Design • Build • Program • Learn • Compete

SERVO

SERVO MAGAZINE

FOR THE ROBOT INNOVATOR
www.servomagazine.com

June 2012

TEAM DARE AND THE ROBOT BAND

U.S. $5.50 CANADA $7.00
06
0 71486 02422 4

HIGH VOLTAGE HIGH QUALITY

From economical sport to high end, ultra premium, Hitec has the perfect high voltage servo for you!

SPORT CLASS

PREMIUM CLASS

HS-430BH
DELUXE BALL BEARING

	6.0 Volts	7.4 Volts
Torque:	57 oz-in	69 oz-in
Speed:	0.16 sec/60°	0.14 sec/60°

HS-5585MH
HV CORELESS METAL GEAR

	6.0 Volts	7.4 Volts
Torque:	194 oz-in	236 oz-in
Speed:	0.17 sec/60°	0.14 sec/60°

HS-5685MH
HIGH TORQUE

	6.0 Volts	7.4 Volts
Torque:	157 oz-in	179 oz-in
Speed:	0.20 sec/60°	0.17 sec/60°

HS-7245MH
HIGH TORQUE CORELESS MINI

	6.0 Volts	7.4 Volts
Torque:	72 oz-in	89 oz-in
Speed:	0.13 sec/60°	0.11 sec/60°

ULTRA PREMIUM CLASS

WATERPROOF CLASS

HS-7950TH
ULTRA TORQUE CORELESS

	6.0 Volts	7.4 Volts
Torque:	403 oz-in	486 oz-in
Speed:	0.17 sec/60°	0.14 sec/60°

HS-7955TG
HIGH TORQUE CORELESS

	4.8 Volts	6.0 Volts
Torque:	250 oz-in	333 oz-in
Speed:	0.19 sec/60°	0.15 sec/60°

HS-M7990TH
MEGA TORQUE HV MAGNETIC ENCODER

	6.0 Volts	7.4 Volts
Torque:	500 oz-in	611 oz-in
Speed:	0.21 sec/60°	0.17 sec/60°

HS-5646WP
WATERPROOF HIGH TORQUE

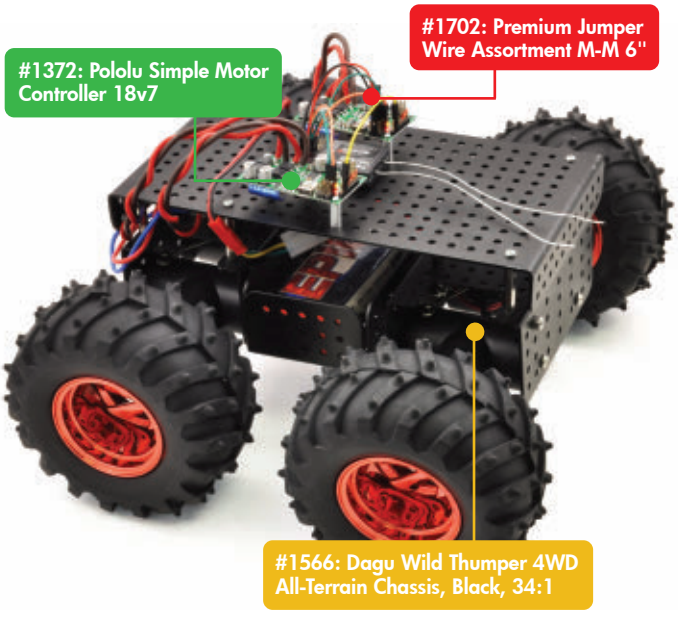
	6.0 Volts	7.4 Volts
Torque:	157 oz-in	179 oz-in
Speed:	0.20 sec/60°	0.18 sec/60°

HITEC

12115 Paine Street • Poway, CA 92064 • 858-748-6948 • www.hitecrcd.com

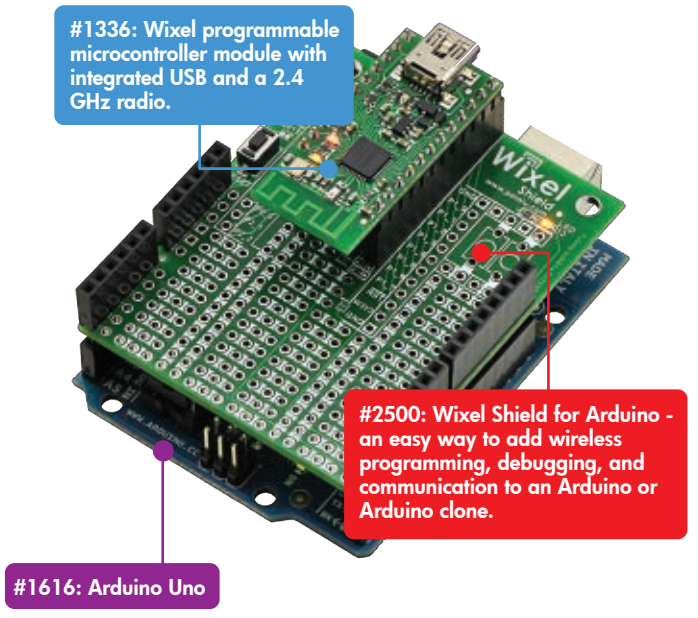
DIY Projects:

Wild Thumper-Based Robot



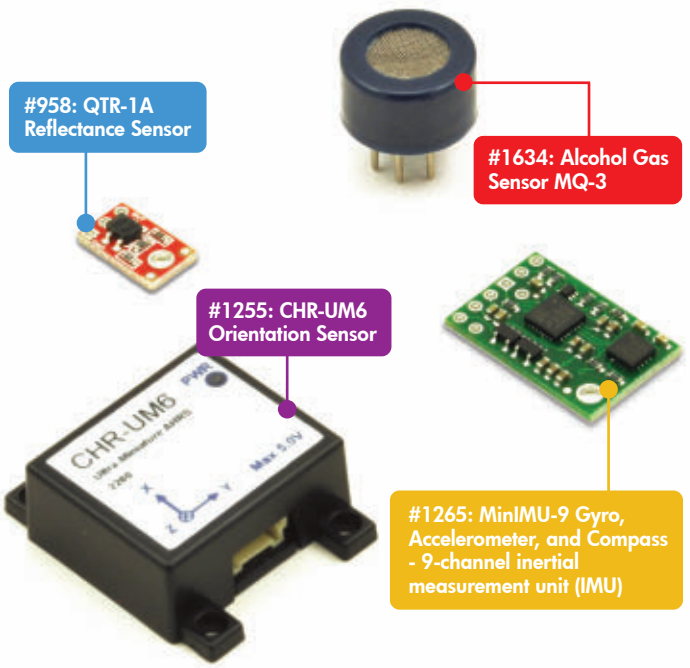
Programmable Controllers:

Wixel and Wixel Shield



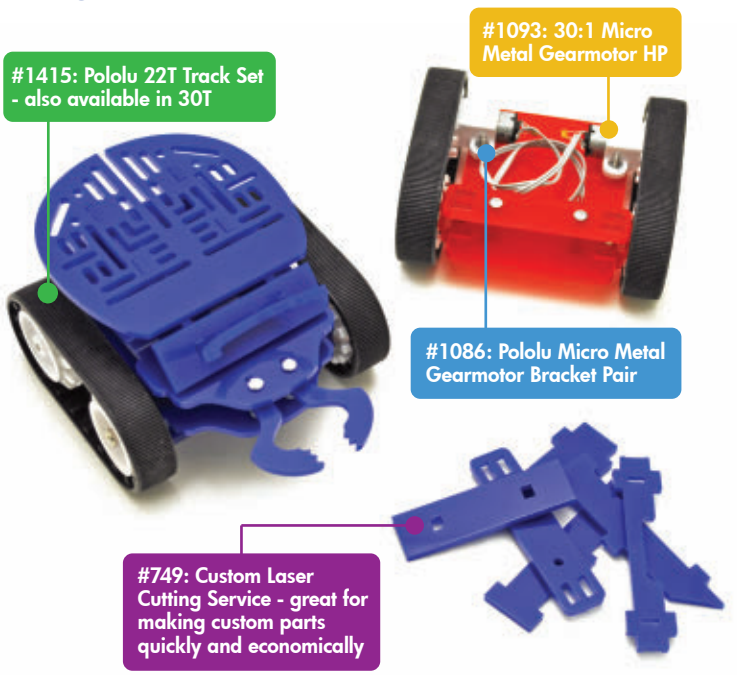
Sensors:

Orientation, Reflectance and More



Custom Laser Cutting:

Design Your Own Chassis



Finding the right parts for your robot can be difficult, but you also don't want to spend all your time reinventing the wheel (or motor controller). That's where we come in: Pololu has the unique products - from actuators to wireless modules - that can help you take your robot from idea to reality.

Pololu
Engage Your Brain

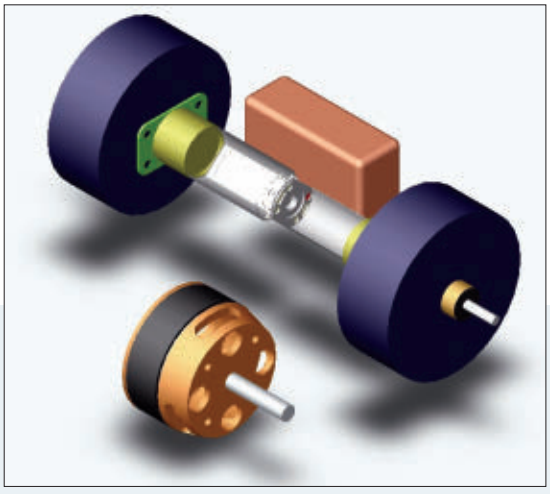
Find these products and more at **www.pololu.com**

SERVO MAGAZINE

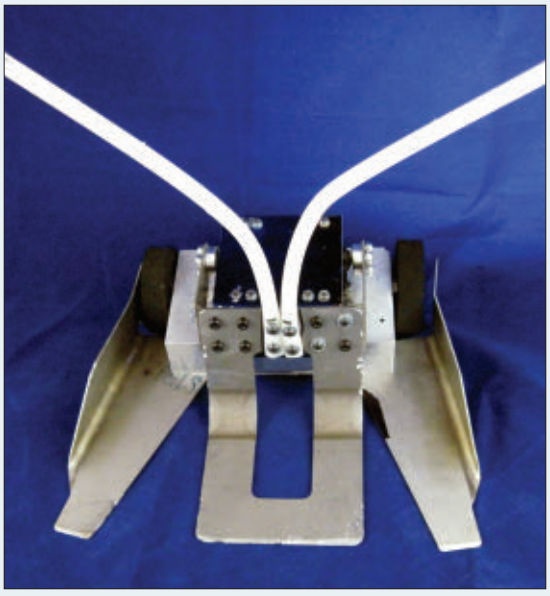
06.2012

VOL. 10 NO. 6

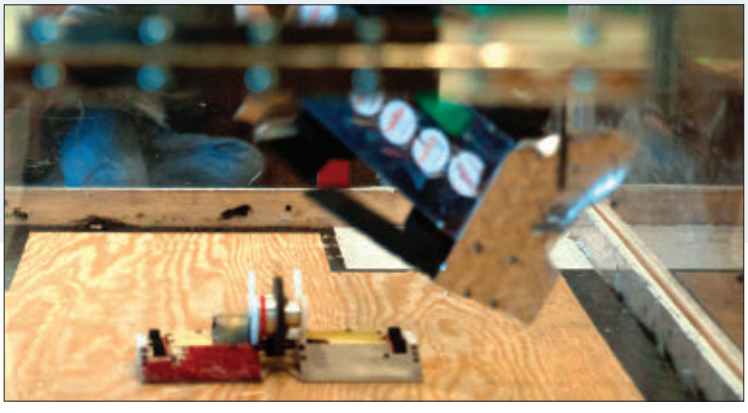
The Combat Zone...



Features

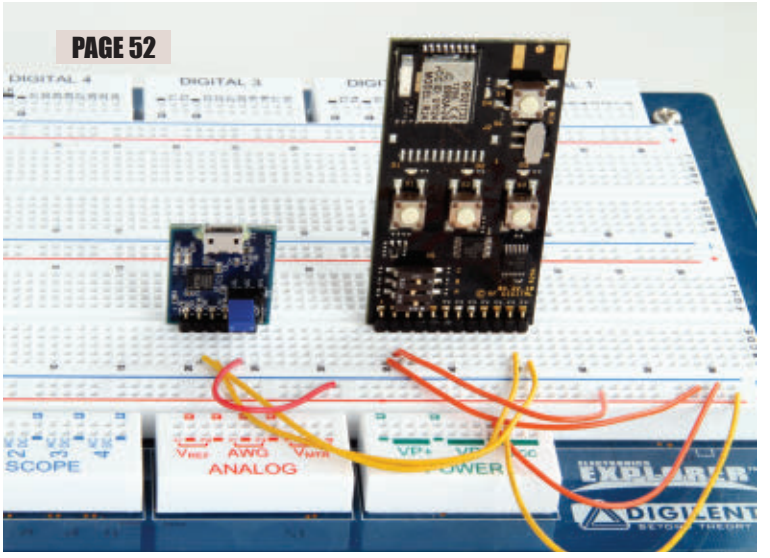


Events

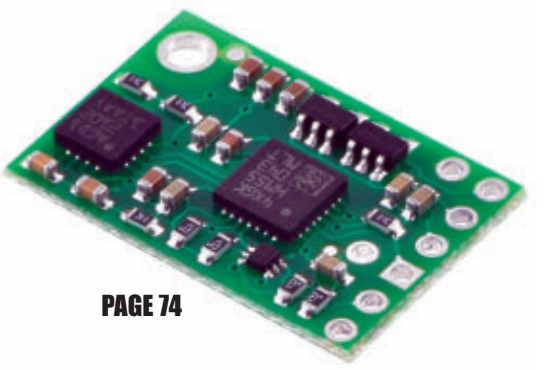


SERVO Magazine **(ISSN 1546-0592/CDN Pub Agree#40702530)** is published monthly for $24.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL ENTRY MAILING OFFICES. POSTMASTER: Send address changes to **SERVO Magazine, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@servomagazine.com

In This Issue ...



Columns



Departments

Mind / Iron

by Bryan Bergeron, Editor

Carpet Roamers Come of Age

Amazon's commitment to acquire Kiva Systems for $775M this past March traveled like a shockwave through the robotics industry. Finally — a commercial application of a mobile robotic platform made the big time. I'm not talking about the occasional mail delivery robot found in a large office building or a pair of drug delivery robots working in the pharmacy wing of a hospital, but swarms of robots in huge warehouses tasked with time-critical transport of everything from books to bowling balls. If the robots prove successful at supporting Amazon's massive volume of orders, then robots from Kiva Systems (or a competing company) are destined for a warehouse near you.

To see these robots in action, take a look at the videos on the Kiva Systems website at **www.kivasystems.com**. The bright orange robots (which resemble the external fuel tanks for outboard motor boats) each support modular warehouse-style racks. Although I haven't seen the official technical specifications, the robots and racks seem capable of supporting at least 100 pounds, as long as the weight is distributed evenly on the shelves of the six foot tall racks.

The videos are worth viewing, if only to illustrate an alternative way for a mobile robot delivery platform to operate. For example, instead of simply picking an item from a shelf and bringing it to a human packer, the Kiva robots bring entire shelves to the packer, in the order in which their contents are needed. Humans stand at predefined spots, unload the shelves, and the robot then moves the shelves to strategic locations in the warehouse. In other words, the warehouse is dynamically reconfigured to suit the current inventory, the needs of the human packers, and the particular orders placed by customers.

According to Kiva marketing statements, moving the shelves of products to stationary human packers instead of having humans hunt through rows of shelves is two to three times faster. The question, of course, is whether that will result in pink slips for two out of three warehouse workers or faster delivery of Amazon products. I suspect the answer is some combination of the two, at least initially. Long-term, I give the edge to the robots and the associated servers and software. Robots don't demand medical care, and don't pass out in the middle of the summer when warehouse temperatures are above 100 degrees.

While I have sympathy for the warehouse workers that might be displaced, I hope the Kiva-Amazon solution is successful. It would portend a bright future for robotics technicians, engineers, and enthusiasts. Consider that every sizeable warehouse will need a technician or two to maintain and repair the fleet of robots and associated computer equipment. Furthermore, companies like Kiva need engineers to continue innovating.

Of course, like other enthusiasts, I'm eagerly awaiting the time when these commercial systems are readily available as affordable commercial platforms on eBay. I'm not sure what I would do with one or two of the beasts — maybe moving furniture around so that my robotic vacuum cleaner can do a better job.

Perhaps you have a better idea? If so, please consider sharing it with our readers. SV

FOR THE ROBOT INNOVATOR SERVO MAGAZINE

Published Monthly By
T & L Publications, Inc.
430 Princeland Ct., Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
Webstore Only **1-800-783-4624**
www.servomagazine.com

Subscriptions
Toll Free **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277, N. Hollywood, CA 91615

PUBLISHER
Larry Lemieux
publisher@servomagazine.com

ASSOCIATE PUBLISHER/ VP OF SALES/MARKETING
Robin Lemieux
display@servomagazine.com

EDITOR
Bryan Bergeron
techedit-servo@yahoo.com

CONTRIBUTING EDITORS
Jeff Eckert, Jenn Eckert, Tom Carroll, David Geer, Dennis Clark, R. Steven Rainwater, Kevin Berry, Gordon McComb, Michael Ferguson, Fred Eady, Pete Smith, Dave Graham, Andrea Suarez, Chris Olin, Morgan Berry

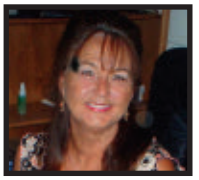
CIRCULATION DEPARTMENT
subscribe@servomagazine.com

MARKETING COORDINATOR WEBSTORE
Brian Kirkpatrick
sales@servomagazine.com

WEB CONTENT
Michael Kaudze
website@servomagazine.com

ADMINISTRATIVE ASSISTANT
Debbie Stauffacher

PRODUCTION/GRAPHICS
Shannon Christensen
Sean Lemieux



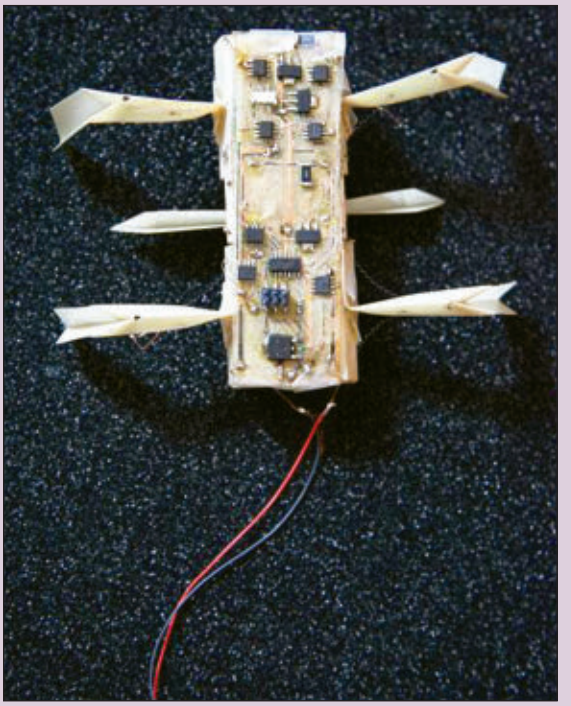
Robytes

Discuss this article in the *SERVO Magazine* forums at http://forum.servomagazine.com.

by Jeff and Jenn Eckert

Create a Custom Bot in a Day

Designing and building a custom robot can be a time-consuming and expensive proposition, but some folks at the MIT Computer Science and Artificial Intelligence Lab (CSAIL, **www.csail.mit.edu**) are envisioning "a whole new way of thinking about the design and manufacturing of robots" that will "make it possible for the average person to design, customize, and print a specialized robot in a matter of hours." The five year project — called "An Expedition in Computing for Compiling Printable Programmable Machines" — also includes researchers from the University of Pennsylvania and Harvard, and is backed by a tidy grant from the National Science Foundation. The basic idea is that you could think of a job that needs to be done in your own household (e.g., swatting flies or picking up your dirty socks), choose an appropriate blueprint from a library of designs, and create the finished bot in a matter of hours. The heart of the concept is 3D printing — an additive manufacturing process in which successive layers of material are laid on top of each other to produce a solid 3D object. Presumably, the fabrication process will not be entirely automated, as it's difficult to imagine generating ICs, batteries, and many other components using a printer. But, hey, they have $10 million and five years to figure it all out. For now, they are focusing on developing a suitable programming interface, writing assembly algorithms, writing a new programming language, and developing new fabrication materials.



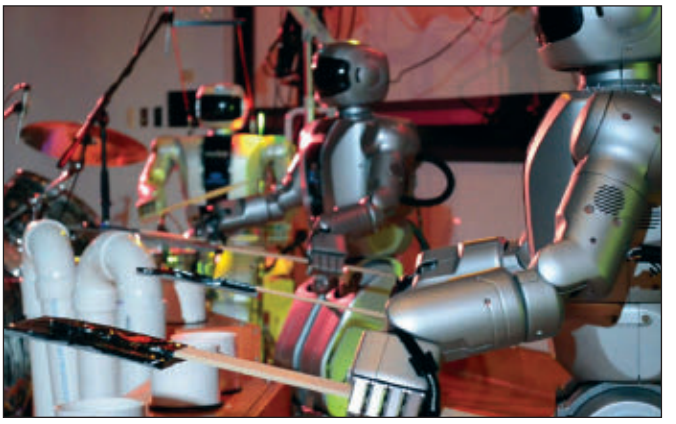
Printable origami insect courtesy of Jason Dorfman, CSAIL/MIT.

Lining up for a turn at the Lely robotic milking machine.

Not Old MacDonald's Farm

One might think that a robotic system named the Astronaut A4 would have something to do with space exploration, but one would be wrong. Mysteriously, it is the moniker for a high-tech milking machine produced by Netherlands-based Lely Group International (**www.lely.com**). This is no ordinary moo-juice extractor, however. No, it's something udderly different (sorry about that). Not only is it designed with the cow's comfort being the prime consideration, it even allows them to mosey on in and be milked whenever they feel like it — even up to four times per day — while the farmer is doing whatever else farmers do. You can even equip the machine with an optional "dynamic feeding module" that automatically changes feed allocations for a particular cow based on the "optimum cost benefit ratio to maximize profits." Apparently, the A4 is a huge success, as Lely recently passed the 12,500 sales mark and is even opening up a new manufacturing facility in Pella, IA, and ramping up its production capacity to 3,000 units annually. For a detailed demonstration, visit **http://www.youtube.com/watch?v=8ONf6DxTnos**. It almost makes you wish you were a cow.

Probably Not Ready for American Idol



"The Hubos" cover a Beatles' tune.

So, if you talked the National Science Foundation (NSF) into buying you seven HUBO advanced humanoid robots — developed at the Korea Advanced Institute of Science and Technology (KAIST) and costing upwards of $400,000 each — how would you harness this resource to benefit humanity? Well, if you were Drexel University's (**www.drexel.edu**) College of Engineering, you would turn them over to the Music and Entertainment Technology Lab (no kidding, it really exists) and use them to create a stage show. Describing the presentation designed to kick off National Engineers Week, lab director Dr. Youngmoo Kim declared, "This is a historic event. Never before have seven adult-sized, fully actuated humanoids appeared on stage together, so it's truly a milestone in robotics research." If you would be amazed to learn that robots can wave, bend their knees, and move (more or less) in time with music, then you don't want to miss the performance, preserved for posterity at **www.youtube.com/watch?v=PLn_BGfD84g**. In a somewhat more entertaining enterprise, four of them bolted from the herd and performed the Beatles' "Come Together" using a drum set and three specially designed "Hubophone" instruments (**www.youtube.com/watch?v=UMQLX-aw_dc**). All pretty amazing, but it sort of makes you wonder what the NSF is doing with the other $7.764 billion in 2012 funding. To apply for your share of it, log onto **www.nsf.gov/funding** and browse the opportunities.

Talk to the Animals

At first, the headline was a little confusing. After all, why would anyone want a robotic version of an Irish singer-songwriter? But then I noticed that we're talking about a bonobo (Pan paniscus) — a little-known great ape that, unlike its cousin the chimp (Pan troglodytes schweinfurthi) but like Bono (Pan U2ibus), is pretty good at walking erect. Not to be confused with Sonnybono (Pan chergonilus), who seldom did. In any event, the Bonobo Hope Great Ape Trust Sanctuary (**www.greatapetrust.org**) in Des Moines is home to eight of them (plus a couple orangutans); these critters can communicate using nearly 400 words as represented by lexigrams on a touch screen. That's where RoboBonobo comes in. The folks at the Trust have created this prototype as a sort of play toy for the bonobos. Using wireless keypads and iPads, they can control it and even make it chase humans around and shoot them with a water cannon (which the bonobos seem to find highly amusing). The problem is, neither the Trust nor the apes have enough money to further develop the program. (Apparently, they've never heard of the National Science Foundation.) For this reason, they're asking for donations to raise the paltry $20,000 needed to create a set of Internet-connected keyboards and apps that will allow both apes and people to communicate with each other. This includes an app that will translate human speech into the appropriate lexigrams. If you want to offer your support, go to **www.kickstarter.com** and search for "bonobo." The best part is that if you pledge at least $500, you'll get to have a live Skype session with a real bonobo. SV

RoboBonobo, a mechanical companion for the real thing.

GEERHEAD

by David Geer

Contact the author at geercom@windstream.net

Discuss this article in the SERVO Magazine forums at http://forum.servomagazine.com

Robot Band DARES to Sound Off in Contest

Last time, we featured a couple of contestants from the ongoing Boca Bearing 25th Anniversary Innovation Contest. The miniature bearing company is celebrating all year by giving away more than $20,000 in prizes and monetary awards to winning entrants.

Contestants include videos of their creations in action. Projects must use one or more ball bearings, roller bearings, linear bearings, or any form of full ceramic or ceramic hybrid bearings to qualify for entry.

To recap, each month a winner is selected based on votes cast at the website. Those fortunate to be selected receive an iPad 2. Boca Bearing will select a grand prize winner and two runners' up from among the monthly winners. While the two finalists will each receive a 3D printer from Makerbot Industries, the grand prize winner will enjoy $10,000 in cash as their reward.

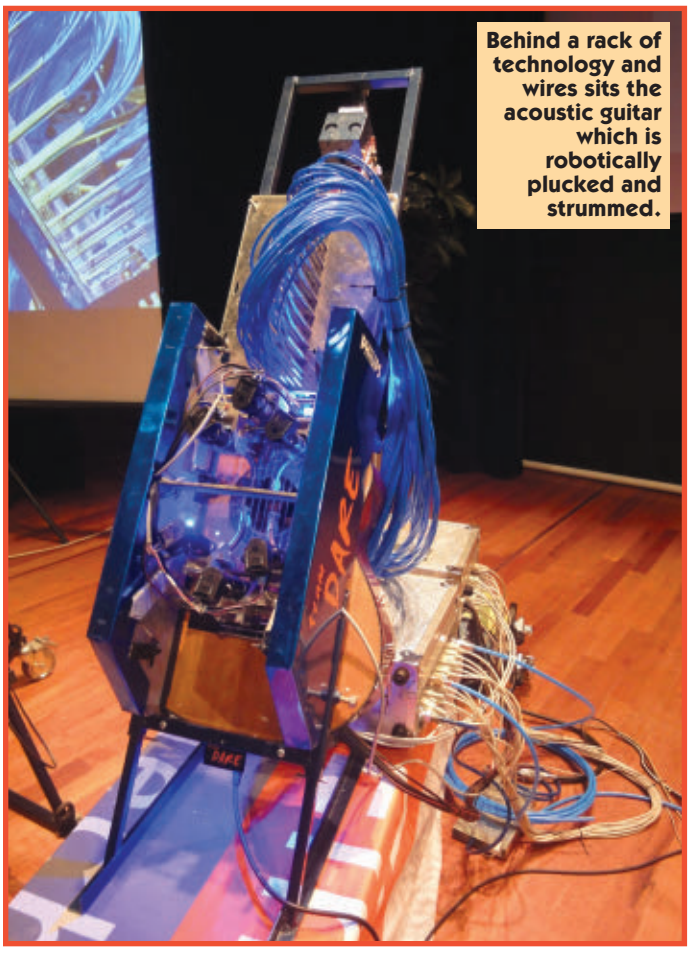
Let's take a look at one more of the entries — a robot band.

The robot band performing live on stage.

A Multi-Instrument Robot Band

TeamDARE Eindhoven are the creators of the robot band. They are a modest group of engineers who love robot building. The team started out in an internship at the TU/e (The University of Technology Eindhoven, the Netherlands) in 2001. Team members originally introduced the robotic trio at Eurobot (an international robotics competition). The team's

The robotic drum kit complete with bass drum, snare, drums, and cymbals.


Behind a rack of technology and wires sits the acoustic guitar which is robotically plucked and strummed.

talents consist collectively in embedded technologies and intelligent systems.

With the help of sponsors, the team has time to work and play at robotics, demonstrating strengths in computer science, mechanics, and electronics. The group is adept at circuitry design and layout, mechanical design and construction, and software design using open source tools.

TeamDARE has been building robots together since before 2002, when they began in the Robot Wars competition, going on to compete in the 2005 and 2006 Eurobot events. By 2008, the team had participated in the Artemis Orchestra Contest which was the seedpod for the robot band of today.

TeamDARE was adamant about creating a robotic band that would be as entertaining as any human counterpart(s), while using acoustic instruments that have not been modified in any way.

The bearings come into play due to the rotating shafts and the 24 electromechanical actuators the team used to actually play the instruments. The mechanics of the robot band are mounted directly onto the motors or gearbox shaft in use whenever possible. This fully applies the load capacity of the motor or the gearbox bearings. These are typically grease lubricated shielded deep-groove bearings.

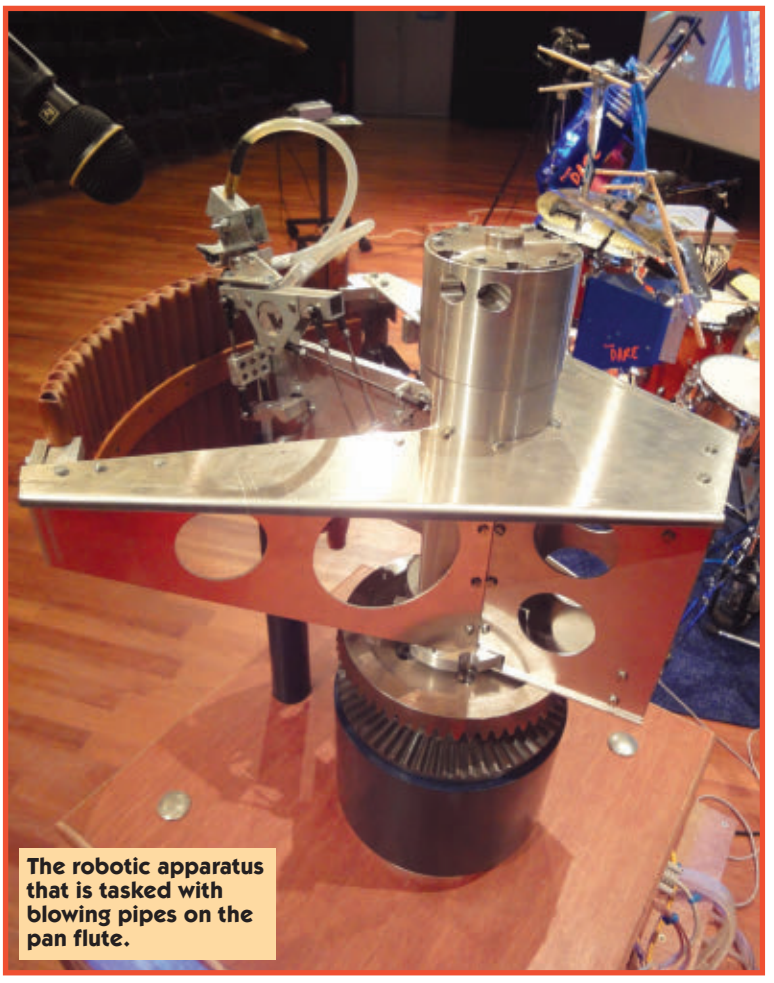
The pan flute provided the most demanding challenges for the application of bearings. The flute's base uses three shafts that are concentric. This makes for a tight squeeze for those bearings. The team used thin section deep-groove ball bearings to satisfy these constraints.

The Robot Band and TeamDARE Indepth

The DARE in the team moniker is actually a clever acronym for Daring Autonomous Robots Eindhoven, which the team has used since 2005. The team is made up of nine members from a variety of high-tech companies and institutes in or near Eindhoven. Companies where team members work include ASML, Philips Research, Prodrive, CCM, DAF Trucks, The Mathworks, and the University of Technology, Eindhoven — which all of the team members attended at some point.

More specifically, the team member's expertise comes from working in lithographics at ASML, corporate research at Philips, electronics design at Prodrive, and mechatronics at CCM.

TeamDARE is rooted in the appearance of some of its members at Eurobot in 2001. Their entry helped to fulfill an internship at the university. The team was comprised of



The robotic apparatus that is tasked with blowing pipes on the pan flute.

students from the departments of electrical engineering, mechanical engineering, and computer science. TeamDARE took second place in their first competition together. In 2005 and 2006 (when the team competed again in the Eurobot), it was supported financially by the university, and by Philips and ASML.

In their careers, TeamDARE members work with complex embedded systems like wafer steppers from ASML (used in the creation of small circuits in a lithographic process) and MRI scanners from Philips. These technologies are constructed using high quality standards in mechatronics and software. The work must also be robust, for example, when engineers make DAF trucks that have many moving parts that must function in challenging environmental conditions.

By applying the same quality standards and craftsmanship, TeamDARE is able to make its robots work right out of the box — even when they must be assembled in high stress conditions such as in dark, loud, and small podiums.

Past Robot Projects

In the Eurobot competitions, teams must typically build two or more robots that will complete a given task autonomously in 90 seconds, while working against an opposing robot. In order to keep the playing field even among new and old competitors, the object of the game is changed year after year. The autonomy and time limits help ensure that the team with the most savvy has the best chance of winning.

When competing in Eurobot 2001, TeamDARE built 11 competition robots. Each robot solved one small part of the overall goal or challenge of the event. The goal that year was placing flags on cylinders of different heights. This team's robot swarm won second place and its concept was so successful that the next year the rules were changed, limiting the number of robots that each team could build for the competition.

After that, the core members of the team entered the Dutch version of RobotWars with a robot named Cyclone. Cyclone was a large moving cylinder with a spinning outer shell. This was quite a feat for a robot that was not much more complex than an R/C car.

For the Artemis Orchestra Contest in 2008, the team built the robotic acoustic guitar that is now part of the robot band today. It took the group about four months to design and construct everything. They continuously expand the robot band with new instruments, waiting for new chances to compete — which came with the Boca Bearing contest.

Circuit Design and Layout

Each musical robot is similar in concept, using a high level circuit design with three layers including actuation, scheduling, and control. At the lowest level, actuation is achieved by using microcontrollers from the AVR32 family from Atmel to control motors and address sensor information. At this level, the controllers have no knowledge of music and only handle simple actuation commands, such as moving a motor from A to B within a given profile.

The intermediate layer schedules the actuations, and also runs on microcontrollers (low-end eight-bit RISC controllers). These controllers receive lists of musical notes for a given instrument, along with instructions for when and how each note should be performed with its exact timing and volume. The system translates these lists of notes into commands that the low level layer executes. Each command is sent at its corresponding time.

At the highest level, the system controls the songs. The system decomposes each song for the instruments that must play it, and for each instrument, the song is converted into the lists of notes that the system sends to the scheduling controllers. The highest layer is no longer time critical and does not have to be real time. This highest layer is easily implemented on a PC.

This layering is important for two reasons. First, there needs to be precise control and actuation, together with an easy user interface that can start, stop, and add songs. Second, it is important to be able to extend the band to an arbitrary number of instruments.

TeamDARE built the circuit boards for this and etched some themselves.

Robotics and Mechanics

With respect to mechanics, each new robot in the band has seen a steady increase in complexity. The guitar is the most simple of the three. It consists of six plectra on little arms that each actuate one string. The arms are rotated with a servomotor.

The left hand of the robotic guitar player consists of a grid of pneumatic pistons that press the string down at the correct fret position. The robot controls the volume of each note by using a second servomotor to control the height of the plectrum with respect to its string.

The drums conceptually seem simple: one stick for each tom, controlled by a servomotor. However, in order to get the timing, volume, and bounce of the drum sticks right, the system necessitated a very complex motion controller. Furthermore, the pedal for the hi-hat and the muting of the crash cymbal were implemented using pneumatics.

The pan flute robot is the most complex of the three in terms of mechanics. Instead of having a nozzle for each tube, the team wanted to use a single pipe and rotate the pan flute back and forth. In order to get the correct tone out of each tube, the nozzle needed to be controlled over three degrees of freedom: horizontally, vertically, and on one axis of rotation.

The airflow influences the pitch of the sound. On each tube of the pan flute, a semitone can be produced in addition to a note. For this, a tube needs to be partially obstructed. The team implemented this using a small rubber cushion that mimics lips and a second airflow supply.

Software and Intelligence

The whole band currently runs on eight microcontrollers and two PCs, due to the layered design. Most of the code on the microcontrollers is written in C, with the exception of some small critical parts that are implemented in Assembly language. The PC software is mostly written in C++. For some nice 3D visualization on a big screen, the team wrote a player in C++ and GPU code which runs on a second PC.

In addition, a lot of software has been written to automate the team's song writing, mostly in C and C++. Basically, these tools convert MIDIs to a format suitable for the robots; they can also simulate how a song will sound when played by the bot band.

Final Note

TeamDARE already has plans in case they win the $10,000. It would be to finish a fourth band member which would be a double bass that is already in progress. SV

Resources

Boca Bearing contest page
www.bocabearings.com/innovation-contest/default.aspx

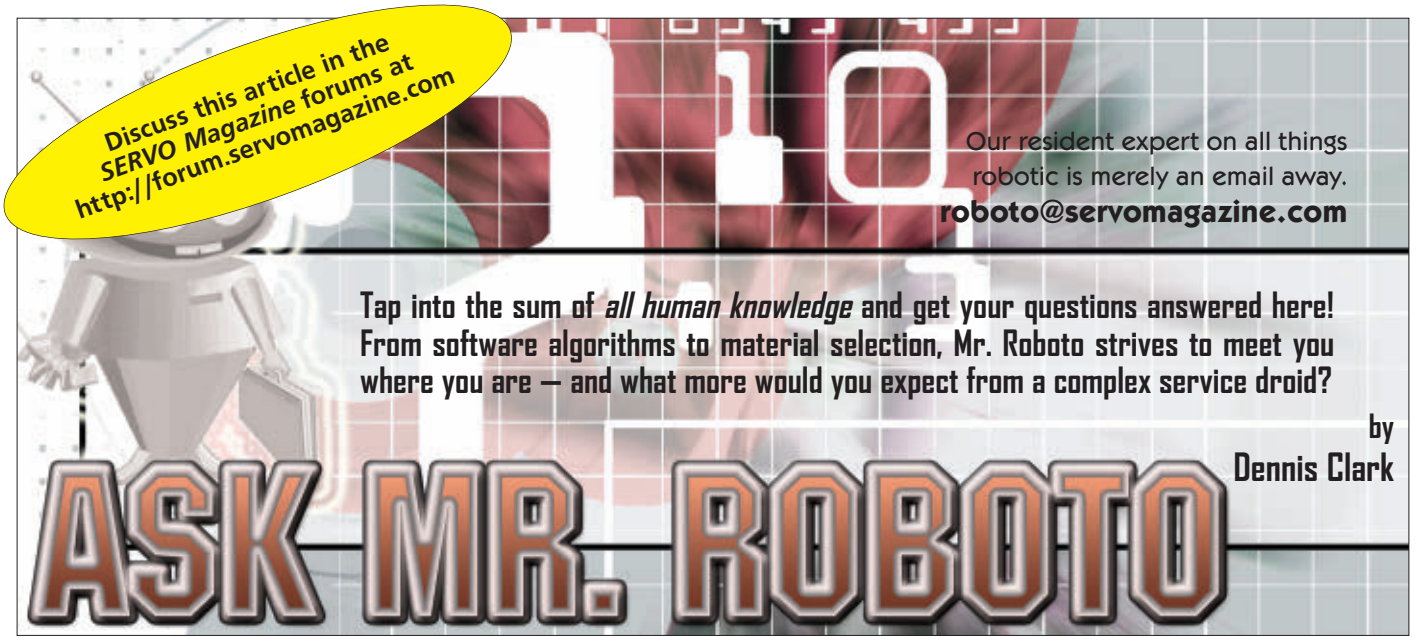
Links to contestants and videos of their projects in action
www.bocabearings.com/innovation-contest/Contestants.aspx

Boca Bearing site
www.bocabearings.com

The Eurobot
www.eurobot.org

TeamDARE sponsors
www.teamdare.nl

Video of TeamDARE robot band performing
www.youtube.com/watch?v=bhBCryKiZ-s&feature=player_embedded



This month, I am continuing with more examples of programming the chipKIT MAX32 board in MPLAB. Of course, I'm going to stick with those things that we robot enthusiasts want to do with our controllers. Let's begin with a different topic.

Q. My question is what sort of programming should I start off with in my telerobotic, and how are algorithms implemented into software for real time feedback?

— Mustafa M.

A. Your first question on what sort of programming should you start off with is something that is asked so many times that there are websites out there with plenty of opinions and options galore. My answer is simply you start with the language (or languages) that you are the most comfortable with. Depending on exactly what you are doing with your tele-presence robot, you may be using more than one language or development environment to handle your job. Because you are using tele-presence, I'm assuming that you will be wanting a network — preferably wireless, but by no means essentially so. To get good response times, you'll want a higher powered controller than a simple line follower robot. Here, once again, we get into comparisons and opinions.

There are lots of choices out there for your robot controller that run higher level operating systems like FreeRTOS, RTLinux, and more. Some small boards (BeagleBoard, Gumstix) will be ARM-based systems running Linux or another OS or RTOS; some Single Board Computers (SBC) will be using "big iron" like Intel processors and will be MUCH higher speed. These latter boards can run full Linux or Windows operating systems.

Once you get into systems running Linux or Windows, then your options open *way* up and you can use your favorite gcc or Studio development environments; the sky is the limit. Because I don't know your application, I can't really suggest anything specific. You can do tele-presence robotics with low cost embedded Linux and RTOS operating systems if your application is simple enough. There are lots of ways to get Internet connections with these low-end boards that have plenty of bandwidth for good response times.

Your second question about algorithms is far more complex. Here, it is time to hit the books. Check our your local library or university for books on mobile robotics. More come out every day, and each builds upon the knowledge of books written the year before. The science is changing so fast that it is difficult to keep up it seems. This question is simply too broad to answer, and the answer would keep changing! Choose a path and follow it. Maybe you'll be writing your own book someday.

Links to some places and products that I mentioned are:

BeagleBoard: **http://beagleboard.org**
Gumstix Boards: **www.gumstix.org**
Rabbit Boards: **www.digi.com/products/wireless-wired-embedded-solutions**

I'm sure your "Google Foo" is as good as mine when it comes to looking up examples of tele-presence robotics, so I won't use up paper with example links. There are a LOT of them!

Q. I belong to a group that is trying to build a robot that will detect flowers. We originally purchased a camera that would take pictures and then detect the flowers in the picture. Our camera idea proved to not work. My team would like to use a tri-colored LED sensor to find our flowers. I am having trouble understanding how this will work. Do you have an idea of how our robot could detect a flower?

— Mason

A. I think that your group was on the right path using a camera to detect flowers. Since you were using a camera, I am assuming that you are using a computer

with sufficient power to analyze the images coming from it. Even still shots from the camera would be enough for a dedicated program to find a certain type or group of types of flowers. There are two programs that come to mind as ideal for your task:

RoboRealm (**www.roborealm.com**): This is a fantastically capable Windows-based program at a very low price. You can set it up to recognize patterns and/or colors very easily. It can be scripted to trigger events, and send messages or even emails. At a hobbyist price of US $59, you really need to try it and get your camera back on your robot.

OpenCV (**http://sourceforge.net/projects/opencv library** or check Wiki at **http://en.wikipedia.org/wiki/OpenCV** to get links to all the manuals and support groups and sites): OpenCV is a set of libraries that Intel released to open source folks that use their hardware to its best advantage so you can write computer vision programs. Unlike RoboRealm — which is a scriptable application for use in lots of computer vision projects — OpenCV is used to write your own computer vision applications. OpenCV can be used on Windows, Mac, or Linux platforms. I have seen some great applications written using this library, and there is a very active user group out there to get help and example code from.

Both of these need some processor power to work, so you're going to be using a Netbook at the very least for your computer. The price of entry isn't that high, so I don't see this as a limiter to your project. Try these two options out with your camera. I think that you'll have success getting your robot to recognize flowers.

Q. I just wanted to reply to a previous issue when you talked about the chipKIT. I have been reading up a good deal about these, and I really want to explore the chipKIT MAX32. I was wondering if you could do a more in-depth write-up on this. I would love to see more of the capabilities of this guy. I know Fred Eady has done some articles on it, but his are more advanced topics; I still consider myself a beginner/novice in the electronics world. Learning about CAN is still well above my head. Now, I'm not saying I want to see a "Hello World" program, but I would like to see some of the more basic stuff such as utilizing the ADC (analog-to-digital converters) with sensors, basic timer/interrupts, and the like. Also, I would like to see it interfaced with MPLAB utilizing C instead of the Arduino processing structure. If I want help with that, I can just go to the Arduino site. I'm more interested in using this as a development board rather than a small projects board.

— Corey Hastings

A. This just happens to be the next installment of my apparently on-going series on programming a Digilent chipKIT MAX32 board using MPLAB 8. This month's follow-on will continue down the path of using a Microchip ICD3 and the MPLAB 8.84 IDE to program a MAX32 to do analog-to-digital sensor reading. I'll be using (as examples) two variable resistors (pots) to adjust the voltage read on two analog ports. You would use this for a Sharp GP2Y0A21 (or similar) range finder.

I'm going to continue to use the Microchip C30 peripheral library routines to set up and use our PIC32 hardware this month as I have in the past. With other PIC processors, I've shown how to directly twiddle the register bits and set up the configurations manually; not so with the PIC32. I've chosen to stay with the peripheral libraries for two reasons.

My first reason is that the PIC32 is a *much* more complex chip than other PIC variants, and the datasheet proliferation along with the rich options mix make this a very difficult chip to learn to use quickly. Quite frankly, we want to program robots, not learn a chip architecture in-depth! All of the configuration registers are 32-bit which means to set bits, you need to do much more creative bit manipulations than you would when using a PIC with eight-bit or even 16-bit registers. Microchip has created these libraries and tuned them to work well. Why should I re-invent the wheel?

My second reason is that the PIC32 runs at 80 MHz and has a *lot* more Flash and RAM than the other PIC processors. The MAX32 has 512K Flash and 128K RAM; I don't feel the need to squeeze out every last instruction cycle and byte of RAM with this much capacity at my disposal. With these two reasons in mind, let's just cut to the chase and make things happen!

Figure 1 shows my MAX32 setup. It is somewhat less

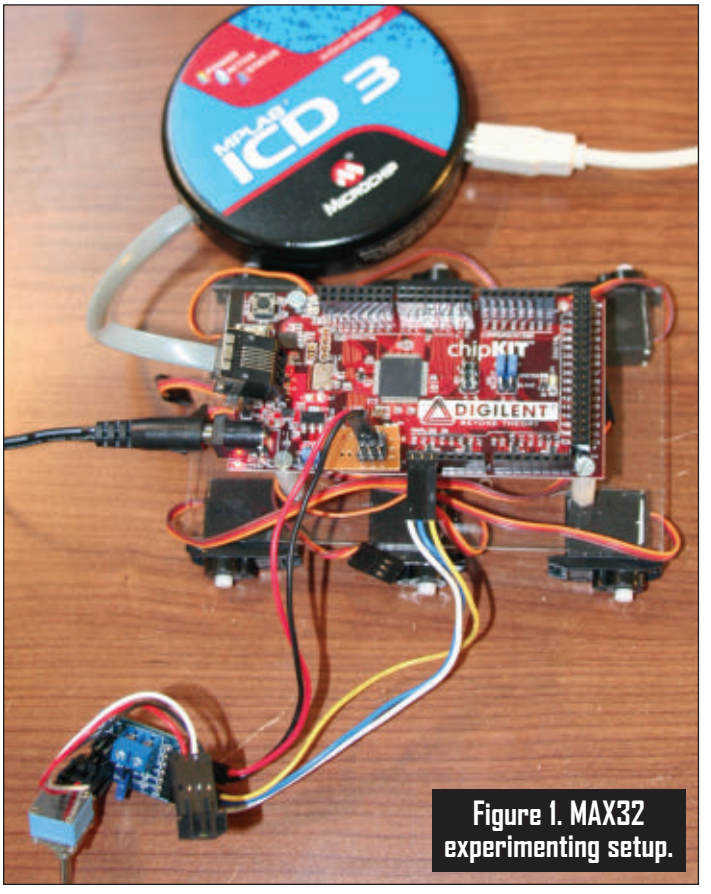


Figure 1. MAX32 experimenting setup.

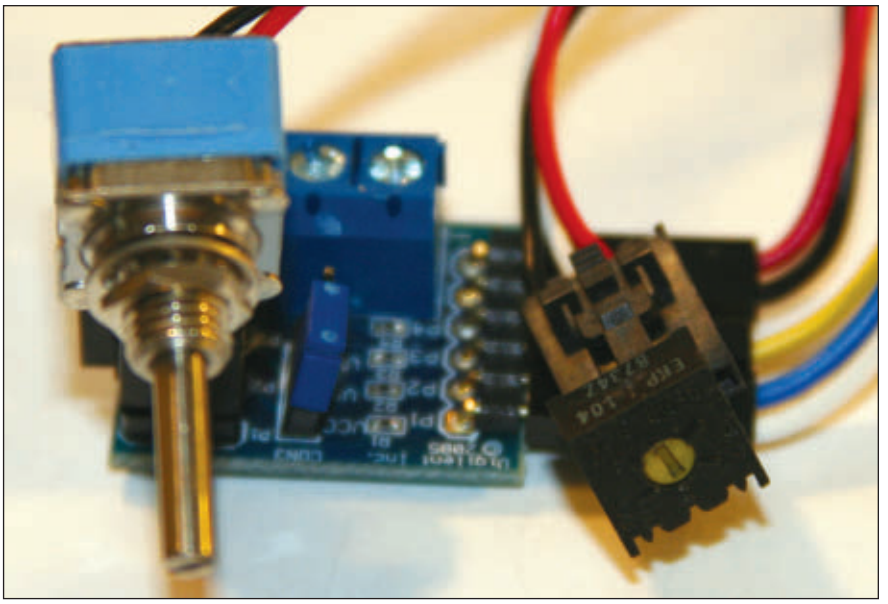


Figure 2. Two pots on a PMOD.

than award-winning for neatness, but it is easy to configure when you are experimenting. (Now you know why so many people develop shields for Arduino(esque) experimenter boards.) I am again using a Digilent CON3 PMOD board for my experiments. I will build on the same code base that I used last month to set up MPLAB and blink an LED; I'll even change the LED blink around to clean up *main()*.

Before I go any further, a discussion is in order about the PIC32 ADC hardware. The MAX32 uses the PIC32MX795F512L which has 16 analog conversion input lines and a multiplexer (MUXA and MUXB) which connects them to a single analog conversion block. The MUX is used so that this ADC block can be either a single-ended ADC input or as a differential input converter. I will be using it as a singled-ended converter in my example code. This means that while it has 16 ADC input lines, it has only one converter, so it can only convert them one at a time.

To make ADC conversions go faster with less processor overhead, the ADC block has 16 result buffers in which to store the conversions. All of these buffers are 32-bit, but you can configure them to report out in 16-bit signed or unsigned and 32-bit signed or unsigned. The signed values simply handle the sign-extend for your math convenience in dealing with the results. There are also signed and unsigned 16- and 32-bit fractional settings, but to be honest, I've no clue how you would use them. If someone understands this, drop me a line. I'd love to hear the explanation! The details of the ADC subsystem are handled not in the PIC32MX795F512L datasheet, but in the DS61104D Section 17, 10-bit A/D converter component-specific datasheet. There are lots of options to play with; I'm going to use only a few in my examples (more on that later).

Figure 2 shows how I connected two randomly selected potentiometers (variable resistors; again pots for short) to the AN0 and AN1 inputs to show you how you can measure more than one analog signal. In my first example, I'll show you how to manually select which one to read. In the second, I'll show you how to have the system automatically scan each configured input and store its value in its own buffer for later use.

Let's look at my first example. **Listing 1** shows how you would set up the PIC32 ADC to manually select between two analog input lines, AN0 and AN1.

The *OpenADC10()* call handles setting the proper registers and their proper bits. *CloseADC10()* should be done to make sure that the ADC is off before it is configured; it will not accept changes properly unless it is off when you change its registers.

Now that we have the ADC configured, let's see how to use it. You may remember from last month I showed how to configure a timer interrupt so we could create a background timer to keep track of milliseconds. We used that to blink an LED. This month, I'll show you a simple state-machine function call that will blink the LED at the rate given by reading our ADC value. This is a simple way to give us feedback that we are indeed changing the ADC input that we want to change. **Listing 2** details those changes and our new *main()* function loop.

The PIC32 ADC is 10-bit which means our range of values is 0-1023. We can blink from really fast to a bit less than 1 Hz. We now blink our LED in the *BlinkLED()* function.

Figure 3. AD1CHS register layout.

Register 17-13: AD1CHS:ADC Input Select Register

R/W-0	r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0
CH0NB	—	—	—	CH0SB<3:0>			
bit 31							bit 24

R/W-0	r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0
CH0NA	—	—	—	CH0SA<3:0>			
bit 23							bit 16

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 15							bit 8

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 7							bit 0

When we call it and our timer has elapsed, we toggle the LED; if we call it and our timer has *not* elapsed, it does nothing. In our main loop, I can set it to choose between AN0 and AN1. I can test this selection by running the code and then turning one or the other to change the LED blink rate.

Yes, I do have to recompile and download each time I want to switch between pots. It isn't the most efficient coding, I admit. As an exercise for the student, write a bit of code and show the electrical connections it would take to flip a switch to decide which analog input to choose.

Note that no matter which analog input we choose, we read from buffer 0. We are manually selecting the analog input with the *SetChanADC10()* function call. Speaking of that call, do you see the (0<<16) value? This is because the MUXA selection is in the upper 16 bits of the AD1CHS register (see **Figure 3**). If we were to programmatically change this value, it is best done this way rather than with the peripheral library defines.

Now, let's see how we could do this using the PIC32 scanning functionality to handle both of our analog input lines without us having to deal with changing them. **Listing 3** shows how we would configure the ADC module to handle this.

Listing 1: Setting up the ADC for manual selection.

```
unsigned long int    c1,c2,c3,cPort,cScan;
    CloseADC10();                                   // make sure it's off.
    /*
    *  Configure the ADC system to use AN0 and AN1, (A0/A1) on the Max32
    * config1: 16bit unsigned 16 bit, auto trigger, auto sample
    * config2: Use AVCC/AVSS reference, No calibration, no scan
    * config3: Use RC ADC clock, use 15 sample periods to acquire
    * configPort: AN0 and AN1
    * configScan: off
    * This setup sets everything up as a manual ADC conversion
    * select manually the channel you want sampled value.
    */
    c1 = ADC_FORMAT_INTG16 | ADC_CLK_AUTO | ADC_AUTO_SAMPLING_ON;
    c2 = ADC_VREF_AVDD_AVSS | ADC_OFFSET_CAL_DISABLE | ADC_SCAN_OFF;
    c3 = ADC_CONV_CLK_INTERNAL_RC | ADC_SAMPLE_TIME_15;
    cPort = ENABLE_AN0_ANA | ENABLE_AN1_ANA;
    cScan = SKIP_SCAN_ALL;
    SetChanADC10(0<<16);
    OpenADC10(c1,c2,c3,cPort,cScan);
    EnableADC10();                                  // Go!
```

Listing 2: Main loop and LED blinker.

```
void BlinkLED(unsigned short rate)
/*
*       Blink "LED4" on the Max32 board by toggling it every
*       "rate" milliseconds.
*/
{
        static delay = 0;

        if (t_1ms > delay) {
                mPORTAToggleBits(BIT_3);
                delay = t_1ms + rate;
        }
}

int main(void)
{
        unsigned short value;
        int alt = 0;

        Init();

        while (1) {
                /*
                *       0 will select AN0, 1 will select AN1
                */
                SetChanADC10(0<<16);
                value = ReadADC10(0);
                BlinkLED(value);
        }
}
```

The c2 configuration word includes the *ADC_SAMPLE_PER_INT_2* define which tells the ADC module that it will be taking two samples to store into the AD1BUF0 and AD1BUF1 buffers. There are 16 buffers and 16 analog input lines. Convenient, isn't it?

Finally, **Listing 4** shows how we use this new analog input setup. Again, I have to change the code, recompile, and download to change which analog channel to use. Again, I offer my readers a challenge to improve my code. Go for it!

I can see you all frowning now and thinking, "Hmph! That isn't very robotic to twist a knob and make an LED blink. I want a sensor!" Okay then. Let's use a Sharp GP2Y0A21 IR range finder to spice up the mix. This device takes 5V to run and outputs a signal whose maximum value is about 3.2V (which is safe for our MAX32 whose PIC32 is powered at 3.3V). Just to be sure, you can put a 100 ohm resistor or so in series with this signal on the off chance that the IR ranger level goes higher than 3.3V. I've never had one do this, but devices sometimes fail, so it would be a prudent move to safeguard your analog line on the MAX32.

Listing 3: Setting up the ADC for scanning channels.

```
unsigned long int       c1,c2,c3,cPort,cScan;
CloseADC10();    // make sure it's off.
/*
*  Configure the ADC system to use AN0/AN1, (A0/A1) on the Max32
* config1: unsigned 16 bit, auto trigger, auto sample
* config2: Use AVCC/AVSS reference, no calibration, scan channels
* config3: Use RC for Tad clock, use 15 sample periods to acquire
* configPort: AN0 and AN1
* configScan: on, scan AN0 and AN1 on every pass.
* This setup scans the configured ADC channels and stores results
* in ADC1BUF0 and ADC1BUF1.
*/
c1 = ADC_FORMAT_INTG16 | ADC_CLK_AUTO | ADC_AUTO_SAMPLING_ON;
c2 = ADC_VREF_AVDD_AVSS | ADC_OFFSET_CAL_DISABLE | ADC_SCAN_ON |
      ADC_SAMPLES_PER_INT_2;
c3 = ADC_CONV_CLK_INTERNAL_RC | ADC_SAMPLE_TIME_15;
cPort = ENABLE_AN0_ANA | ENABLE_AN1_ANA;
cScan = 0xFFFC;          // OpenADC10 inverts or "skips" these
SetChanADC10(0<<16);     // Not needed, always uses MUXA
OpenADC10(c1,c2,c3,cPort,cScan);
EnableADC10();           // Go!
```

Listing 4: Scanning the ADC mode usage in main loop.

```
int main(void)
{
       unsigned short value;
       int alt = 0;

       Init();

       while (1) {
              /*
              *      In "scan "mode, 0 is ADC1BUF0 = AN0 and
              *      1 is ADC1BUF1 = AN1.  No need to select
              *      a channel, both are stored for use.
              */
              value = ReadADC10(0);
              BlinkLED(value);
       }
}
```

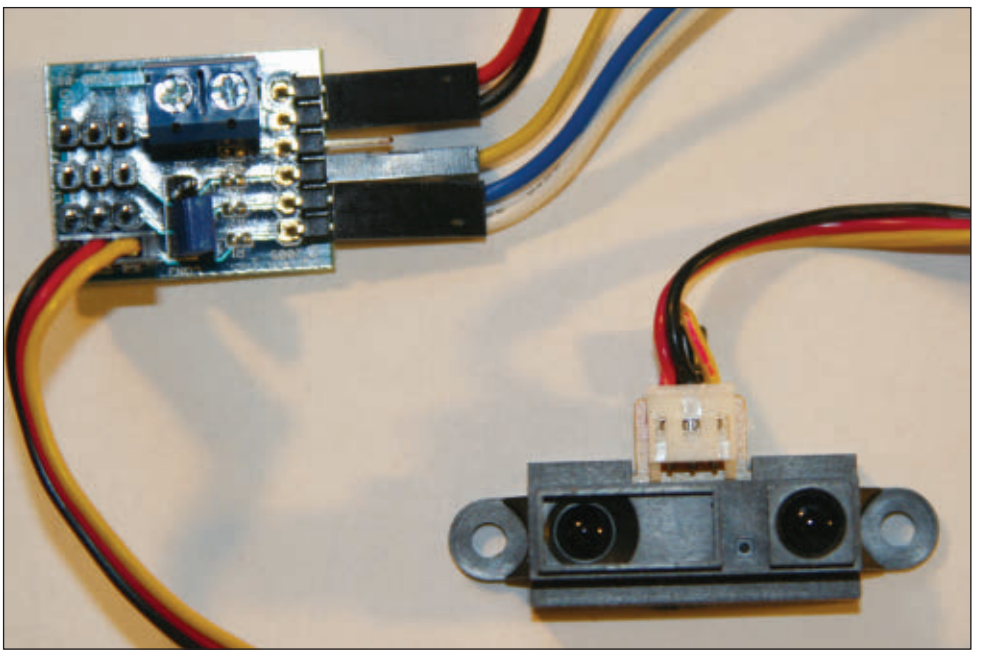


Figure 4. Sharp IR range finder setting the LED Flash rate.

Figure 4 shows my connection to the CON3 PMOD board. I set my connectors up on the IR range finder to match the servo-style power and signal order used on this PMOD (and on lots of other controller boards), so it is as easy as plugging it in to use now that we have our analog system up and running.

Using the code here, you can now control the Flash rate by moving things closer and/or farther from the range finder. This model has about an 80 cm range.

Remember these devices output a lower voltage as you get farther away, and their maximum output is when you are a few centimeters away from the lenses. Here is another exercise you can do! Find the article in *SERVO Magazine* I wrote that gave you an Excel spreadsheet that shows you how to convert these voltage readings directly to distance. You can get the theory from the Acroname website at **www.acroname.com**.

Sharp IR range finders are notoriously noisy electronics and pull quite a bit of current. I've found that adding an electrolytic cap of 10 µf or so directly on the board's power leads will do wonders for cleaning up your microcontroller's power bus. Look at **Figure 5** to see what I mean.

If you look at your power bus on a 'scope while one of these sensors is running, you'll be amazed at the electrical noise they generate. You'll do the rest of your analog circuitry a favor by filtering this device's power line.

Okay, we're done for now. Go experiment. It's easy to spend hours with things like this, but don't feel bad! You're learning and working on gaining new knowledge, so it is time well spent.

I'll repeat some document numbers you can get from the Microchip website to learn about the MAX32 processor:

- DS61156G: PIC32MX795F512L.
- DS61104D: Section 17 10-bit A/D Converter.
- Under the MPLAB C32 download pages, you'll also want to get the PIC32 peripheral libraries for the MPLAB C32 compiler.

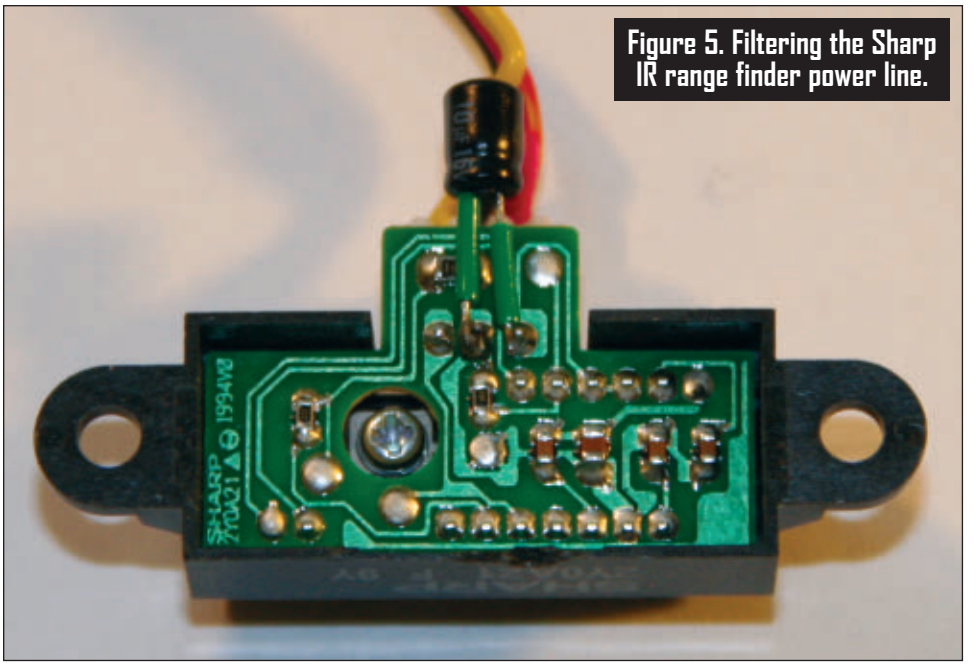


Figure 5. Filtering the Sharp IR range finder power line.

At the article link, you'll find a zip file called robotomay.zip, in honor of the first time I did code for the MAX32. It also has all of this month's code in there and ready for you to use. There is a *#define _MANUAL_ADC_* flag that will allow you to simply flip back and forth between the manual analog select and the scanned analog system.

That's it for this month. Our next installment will include pulse width measurement with input capture for using an inexpensive SONAR, the HC-SR04, and some more sophisticated ways to output data in a more human readable format.

Until then, keep sending me those cards and letters, or emails to me at roboto@servomagazine.com and I'll do my best to answer them! SV

EVENTS Calendar

ROBOTS.NET

Send updates, new listings, corrections, complaints, and suggestions to: steve@ncc.com or FAX 972-404-0269

Know of any robot competitions I've missed? Is your local school or robot group planning a contest? Send an email to steve@ncc.com and tell me about it. Be sure to include the date and location of your contest. If you have a website with contest info, send along the URL as well, so we can tell everyone else about it.

For last-minute updates and changes, you can always find the most recent version of the Robot Competition FAQ at Robots.net: **http://robots.net/rcfaq.html**.

— R. Steven Rainwater

JUNE

2 **Los Angeles/Orange County Robot Challenge**
Santa Ana High School, Santa Ana, CA
A student robot competition for small (20 x 15 cm) autonomous robots.
www.sahsrobotics.org

2 **SRS Robothon**
Center House, Seattle Center, Seattle, WA
This year's competition includes line following, line maze, Mini Sumo, and SRS Robo-Magellan.
www.robothon.org

8-10 **National Underwater Robotics Challenge (NURC)**
Chandler, AZ
NURC teams play out an X-Files case as the Lone Gunmen, sending an ROV robot into Lake Okoboji to search for UFOs while Mulder and Scully distract the authorities.
www.h2orobots.org

8-10 **Robocore**
Jaguariuna, Brazil
Events include Sumo, line following, bot hockey, robot trecking, and remote control vehicle combat.
www.robocore.net/modules.php?name=GR_Eventos

8-11 **AUVS International Ground Robotics Competition**
Oakland University, Rochester, MI
Autonomous ground robots must navigate an outdoor obstacle course within a prescribed time while staying within a 5 mph speed limit.
www.igvc.org

16 **SparkFun Autonomous Vehicle Competition**
SparkFun parking lot, Boulder, CO
Autonomous robots must circumnavigate around the SparkFun building by either ground or air.
www.sparkfun.com

16 **UK National Micromouse Competition**
Birmingham, UK
Robot mice compete to win the coveted Brass Cheese award.
www.tic.ac.uk/micromouse

18-24 **RoboCup Robot Soccer World Cup**
Mexico City, Mexico
All the usual RoboCup events including RoboCup Soccer, RoboCup Rescue, RoboCup@Home, and RoboCup Junior. There will also be a new event called the Festo Logistics Competition that uses a standardized mobile robot platform called Robotino.
www.robocup2012.org

21-23 **MATE ROV Competition**
Orlando, FL
Championship event for high school and college ROV teams.
www.materover.org

23-24 **International Autonomous Robot Contest**
San Diego, CA
Autonomous robots must navigate a course with fixed obstacles.
www.iaroc.org

JULY

3-6 **International Micro Air Vehicle Competition**
Braunschweig, Germany

Various events for autonomous and remote control flying micro-robots.
www.imav2012.org

10-15 **WCCI Competition**
Brisbane, Australia
Events include Human-like Bots, Physical Traveling Salesman Problem, and the Turing Test Track.
www.wcci2010.org

17-22 **International RoboSub Competition**
SSC Pacific TRANSDEC, San Diego, CA
Student teams build autonomous robot subs to accomplish a mission that's different each year.
www.robosub.org

18-22 **Botball National Tournament**
Honolulu, HI
Student teams build autonomous robots that move black and white balls around on a game board.
www.botball.org

22-26 **AAAI Mobile Robot Competition**
Toronto, Ontario, Canada
Different mobile robot contest events each year.
www.aaai.org/Conferences/AAAI/aaai12.php

29 **ASABE Robotics Competition**
Dallas, TX
Student-built agricultural robots compete on a cattle feedlot course. Contest runs through August 1.
http://abe-research.illinois.edu/ASABERobotics

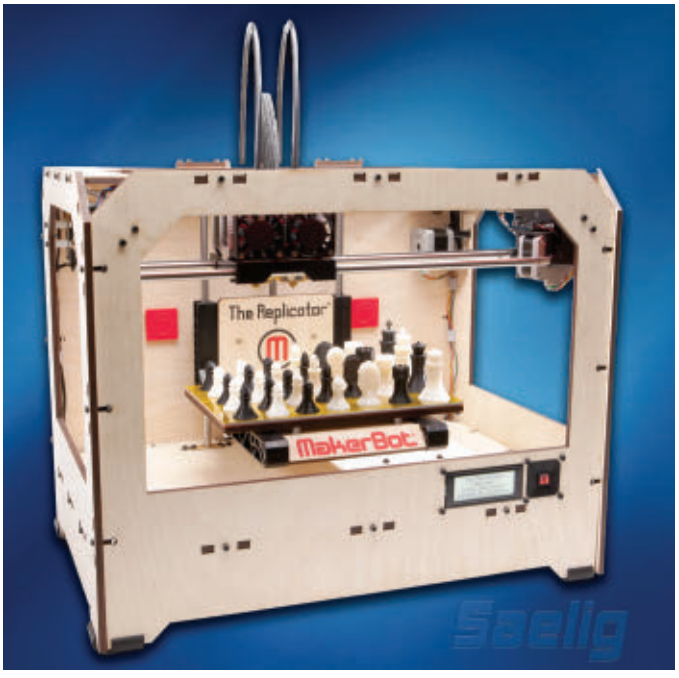
31 **AUVS International Aerial Robotics Competition**
Betty Engelstad Sioux Center, UND
Grand Forks, ND
University teams build flying autonomous robots and sub-vehicles that compete in a task that varies each year.
http://iarc.angel-strike.com

Robotics Showcase

NEW PRODUCTS

Affordable Prototyping 3D Printer

Saelig Company, Inc.,announces the availability of the Replicator™ – an affordable, personal 3D printer offering one- or two-color "printing" of solid objects. The Replicator runs open source 3D printing code and is compact enough to fit on a desktop. Ready within minutes to start printing right out of the box, the Replicator fabricator turns raw feedstock – such as ABS or PLA – into instant prototypes as large as a loaf of bread.



The Replicator is a precision-made parts fabricator built with linear ball bearings and precision-ground 8 mm shafts. It's ideal for personalized manufacturing or prototyping, providing a new way to fabricate designs and variants quickly, as large as 225 x145 x150 mm (8.9" x 5.7" x 5.9"). The Replicator is available with single or dual extruders, facilitating simultaneous two-color printing.

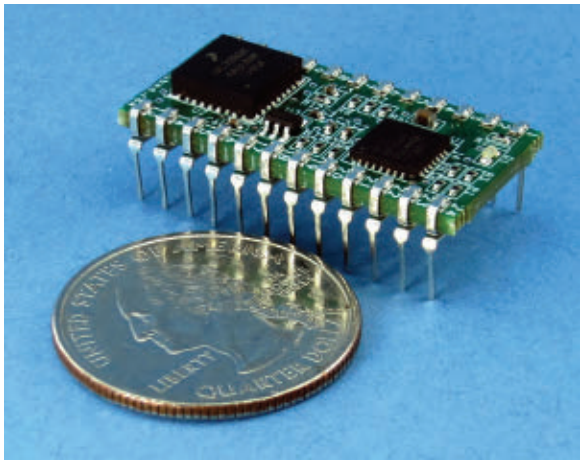
The Replicator features a 4x20 character LCD panel and multi-directional control pad. The LCD screen provides build data as well as monitoring information, and full machine control is possible without the use of a computer. Using an SD card slot or USB connection, model designs can be loaded and built directly from control pad commands. Professional engineers can now quickly fabricate solid objects using tools like AutoCAD and SolidWorks, producing STL or gcode files. ReplicatorG software provided (Linux, Windows, and OSX compatible) enables rapid prototype production. Layer thickness may be selected from 0.2-0.3 mm with the stock 0.4 mm nozzle; parts are built at a speed of 40 mm/s, with positioning precision of 2.5 microns (Z axis) and 11 microns (X-Y axes). Sized for almost any desktop (320 x 467 x 381 mm; 12.6" x 18.4" x 15"), the Replicator weighs 26-29 lbs (single/dual).

For further information, please contact:

Saelig Website: **www.saelig.com**

Miniaturized Motion Controllers

Solutions Cubed, LLC has introduced a new line of miniaturized motion controllers. The Synaptron Micro motion controller is packaged in a tiny 24-pin DIP module. It controls DC motors powered by 6-24 VDC and can handle a current load of 1.5A continuous (5A peak). With the Synaptron Micro, users can employ motion control anywhere.



The module is ideal for open loop motor PWM drive controls or closed-loop motion control. Its vast array of user settings allows it to accept control inputs from serial commands, pulses, or analog signals. Configurable feedback options include analog signals, pulse or frequency signals, quadrature encoders, or internal current or velocity calculations.

Free test software simplifies user customization of control and feedback signals, PID programming, limits switch settings (including virtual limit switches), PWM limits, current limits, and a host of other adjustments. The module is priced at $50 per unit, and $40 in quantities of 250.

For further information, please contact:

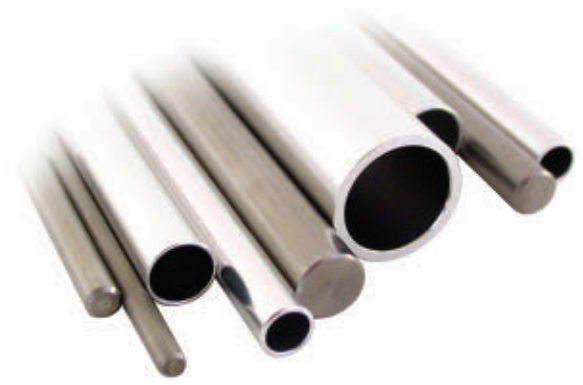
Solutions Cubed Website: **www.solutions-cubed.com**

32 Pitch Aluminum Hub Gears

ServoCity is now offering a new line of heavy duty 32 pitch aluminum hub gears. These gears are machined from 7075-T6 aluminum with a 1/4" thick face to ensure that they will handle demanding applications. The interior face of the gears have been machined in order to save weight without sacrificing durability. These 32 pitch gears are available in a large range of sizes and are designed to attach to aluminum hubs which ServoCity offers in various bore sizes. The hub gears are perfect for robotic applications, pan and tilt mechanisms, camera sliders, and animatronics applications.

Stainless Steel Shafting

ServoCity is also offering a larger selection of precision stainless steel shafting. In addition to their current diameter sizes of 1/8", 3/16", 1/4", 5/16", and 3/8", they have recently added 1/2" and hollow bore shafting.



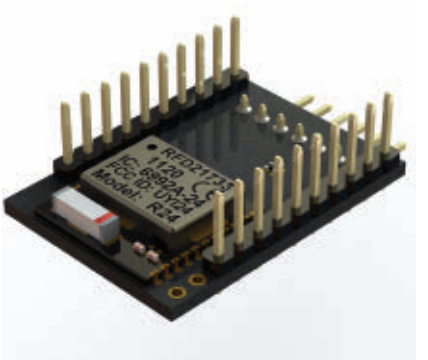
All sizes are available in lengths ranging from 1–12 inches. These shafts are made from 303 stainless steel for superior strength and have a Rockwell hardness of 83B with a 10 RMS micron rated finish. All shafts are cut with a chamfer at each end (.02" x 45 degrees). They are sold in single units starting at just $1.85.

For further information, please contact:

ServoCity Website: **www.servocity.com**

Is your product innovative, less expensive, more functional, or just plain cool? If you have a new product that you would like us to run in our *New Products* section, please email a short description (300-500 words) and a photo of your product to: newproducts@servomagazine.com.

Wireless Application Modules



New instant wireless application modules from RF Digital enable users to build wireless applications instantly, just by writing code for their favorite controller with their existing development environment. Wireless applications can be up and running within minutes since everything is included in one small size package, ready to use.

The modules work with the Atmel ATMEGA164, the Microchip PIC18F44K22, the Texas Instruments MSP430F55, and the Silicon Labs C8051F586.

There's no need for additional development tools or special connectors. No wiring or soldering is needed and RF knowledge is not necessary. You can reuse existing code, and there is no need to fabricate PCBs. All bypass caps are included and the modules plug into any breadboard. They are also compliance approved and fully tested.

Sample code is supplied for Atmel, Microchip, Texas Instruments, and Silicon Labs controllers.

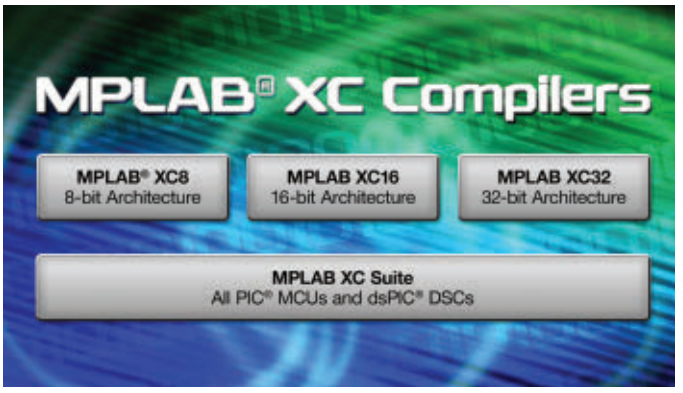
Users simply output serial bytes (using their favorite controller) and the module takes care of the rest. It will seamlessly transfer data over the air up to 500 feet (150 meters).

For further information, please contact:

RF Digital Website: **www.rfdigital.com**

Simplified C Compiler Line

Microchip Technology, Inc., has now introduced MPLAB®XC – its simplified line of C compilers that provide great execution speed and code size for all ~900 PIC microcontrollers (MCUs) and dsPIC® Digital Signal Controllers (DSCs). The MPLAB XC8, XC16, and XC32 compilers offer reduced complexity for eight-, 16-, and 32-bit designers, with three cost-effective optimization levels: free, standard, and pro. The pro editions can be evaluated for free for 60 days. Additionally, MPLAB XC provides support for the Linux, Mac OS®, and Windows® operating systems, enabling designers to use their platform



of choice for embedded development.

Another important consideration for today's designers is the ability to re-use their code and easily migrate to the level of microcontroller performance and features that best suit the needs of each project. MPLAB XC makes it simple to move code from any of Microchip's existing compilers. Additionally, MPLAB XC completes Microchip's tool chain of compatible compilers and debugger/programmers that operate seamlessly within the universal, cross-platform, and open source MPLAB X integrated development environment, reducing both learning curves and tool investments. MPLAB XC compilers are also compatible with the legacy MPLAB IDE.

For further information, please contact:

Microchip Website: **www.microchip.com**

Position Indicator

Firgelli Technologies' new position indicator provides remote visual feedback for all of their feedback type actuators. This is a simple way to add a visual display to an actuator system without modification of the existing design or the addition of a custom controller or software.

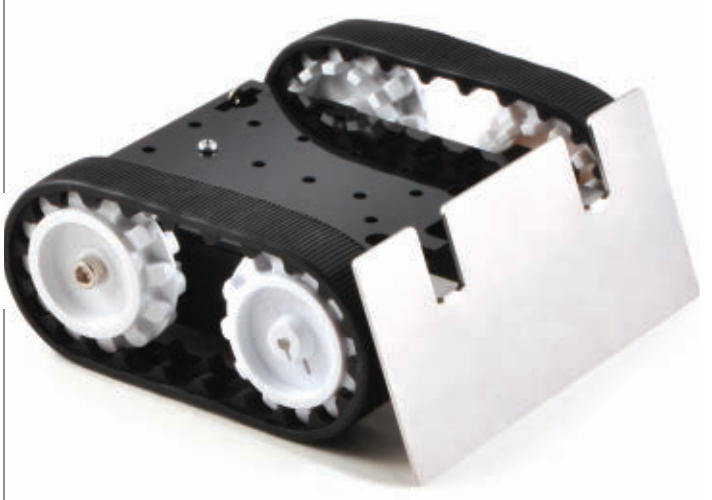
Simply connect power, ground, and the feedback line to the position indicator and it will begin to operate. This indicator is compatible with all –I and –P actuators, as well as the –P/LAC combination.

For further information, please contact:

Firgelli Technologies Website: **www.firgelli.com**

Zumo Chassis Kit

Pololu announces the release of its Zumo chassis — a small, low-profile tracked robot platform. The main body is composed of ABS plastic and features sockets for two micro metal gearmotors and a compartment for four AA batteries (motors and batteries sold separately). The micro metal gearmotors are available in a wide range of gear ratios, making it possible to select the ones that have the best blend of torque and speed for the particular Zumo application. The chassis ships as a kit and — along with the main body — includes two silicone tracks, two drive and two idler sprockets, an acrylic mounting plate, and mounting hardware.



With dimensions of 98 x 86 mm, the Zumo can qualify for mini Sumo competitions. For such applications, Pololu separately offers a basic stainless steel Sumo blade which can be mounted to the front of the Zumo chassis to push around other objects (such as other mini Sumo robots). The design file for the Sumo blade is publicly available and can be used as a starting point for custom laser-cut blades.

The Zumo chassis is available for $19.95.

For further information, please contact:

Pololu Corporation Website: **www.pololu.com**

Lynxmotion

The Lynxmotion Servo Erector Set
Imagine it... Build it... Control it!

Biped Nick

Biped Pete

Featured Robot

The A-Pod is here...
Get yours now!

Youtube videos
User: Robots7

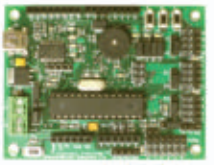
Biped Scout

Biped 209

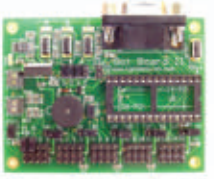
Walking Stick

With our popular Servo Erector Set you can easily build and control the robot of your dreams!

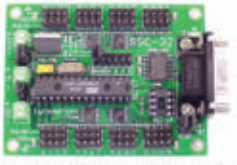
Our interchangeable aluminum brackets, hubs, and tubing make the ultimate in precision mechanical assemblies possible.



New! Botboarduino - $34.95
Deumilanove AT328 compatible.
Lynx footprint with Arduino shield.
USB Programming port.
Speaker, Buttons and LEDs.
Program in Arduino C.
Servo and Logic power inputs.



Bot Board II - $24.95
Carrier for Atom / Pro, BS2, etc.
5vdc 250mA LDO Regulator.
Buffered Speaker.
3 Push button / LED interface.
Sony PS2 game controller port.
Servo and Logic power inputs.



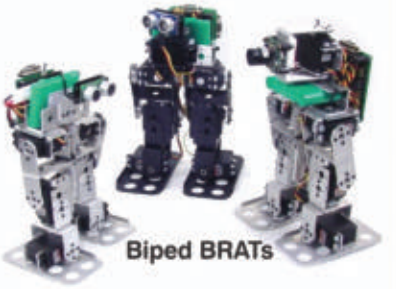
SSC-32 - $39.95
32 Channel Servo Controller.
Speed, Timed, or Group moves.
5vdc 250mA LDO Regulator.
TTL or RS-232 Serial Comms.
Servo and Logic power inputs.
No better SSC value anywhere!

We also carry motors, wheels, hubs, batteries, chargers, servos, sensors, RC radios, pillow blocks, hardware, etc!

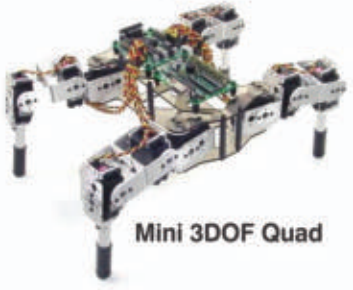
Visit our huge website to see our complete line of robots, electronics, and mechanical components.

CH3-R

T-Hex



Biped BRATs



Mini 3DOF Quad

Phoenix

Images represent a fraction of what can be made!

www.lynxmotion.com

The SES now has over 200 unique components!

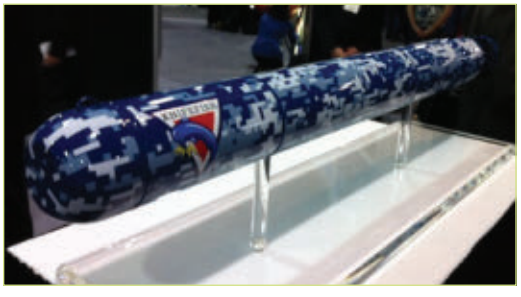
A ROBOT IN THE HAND IS ...

There are lots of innovative ways to land robotic aircraft, from cables to parachutes to controlled crashes. None of these ways are especially ideal, with ideal referring to an aircraft that makes a gentle landing exactly where you want it.

Part of what makes *this* particular robot (under development at the Department of Aerospace Engineering at the University of Illinois at Urbana-Champaign) so cool is the fact that it uses flapping wings for extra maneuverability and probably a little bit of thrust. This bio-inspired model (based on birds and bats) can reorient its wings while gliding, providing glide-phase control without a bunch of extra complicated and heavy actuators. It's highly effective control, too, and allows a thrown micro air vehicle (MAV) to make a pinpoint landing on the back of an outstretched hand.

Getting robots to perch isn't a new idea, although this level of control certainly is. In the past, there have been small robotic aircraft from Stanford's Biomimetics Lab that perform similar pitch-up stall-type maneuvers to perch on vertical surfaces using little claws, as well as planes that can perch on wires from MIT. And, of course, there's that MAV from EPFL that just smashes head first into whatever it wants to land on and then sticks. However, the type of perching that the UIUC team has come up with seems to be a more versatile way to do it, since the MAV is presumably capable of landing on more or less anything — just like a bird.

You can get this same type of precision performance out of rotorcraft, but you don't get anywhere close to the level of endurance that fixed wing aircraft offer, which is why this is potentially an ideal solution: long cruise times combined with pinpoint landings. If they can get this thing to take off again, they'll have it made.



MAC THE KNIFEFISH

So, what's this torpedo looking thingy? It's what the Navy is hoping will be its next-gen anti-mine robot, called the Knifefish. The Knifefish is going to be a 20 foot, 3,000 pound robot that will go on 16 hour missions, snooping around for sea mines using low frequency bandwidth synthetic aperture sonar that's so effective it can even penetrate beneath the ocean floor, depending on the makeup of the sediment (according to Navy Capt. Duane Ashton of the Navy's Maritime Systems Program Office).

The Knifefish is designed to prowl the seas autonomously and can scan "high-clutter" environments using its sonar to tell the difference between all sorts of sea mines and random debris. The only downside for now is that the Fish has to come back to its mothership to upload the information that it vacuums up on its missions.

The torpedo-looking minehunters — developed by General Dynamics — will be in service by 2017, according to Ashton.

EXO-TREMITY

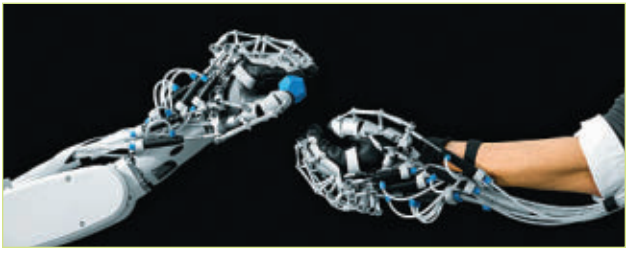
Festo — who is famous for its SmartBird robotic seagull and elephant trunk manipulators, among other things — has unveiled its latest bionic contraption: the ExoHand. The ExoHand is an exoskeleton glove that you can wear to teleoperate a separate robot hand in real time. The cool part is that the device — powered by eight pneumatic actuators — can also be used to make your hand stronger and reduce fatigue during repetitive tasks. Festo says the ExoHand could find applications in manufacturing and medical therapy.

The fingers can be actively moved and their strength amplified; the operator's hand movements are registered and transmitted to the robotic hand in real time. The objectives are to enhance the strength and endurance of the human hand, to extend a human's scope of action, and to help secure an independent lifestyle even at an advanced age.

The ExoHand could also provide assistance in the form of force amplification in connection with monotonous and strenuous activities in industrial assembly, for example, or in remote manipulation in hazardous environments. With force feedback, the human operator feels what the robot grasps and can thus grip and manipulate objects from a safe distance without having to touch them.

Due to the yielding capacity of its pneumatic components, the ExoHand offers potential in the field of service robotics. In the rehabilitation of stroke patients, for example, it could be used as an active manual orthosis.

The exoskeleton supports the human hand from the outside and reproduces the physiological degrees of freedom – the scope of movement resulting from the geometry of the joints.



Eight double-acting pneumatic actuators move the fingers so that they can be opened and closed. For this purpose, non-linear control algorithms are implemented on a CoDeSys-compliant controller, which thus allows precise orientation of the individual finger joints. The forces, angles, and positions of the fingers are tracked by sensors.

NOBODY'S BUDDY

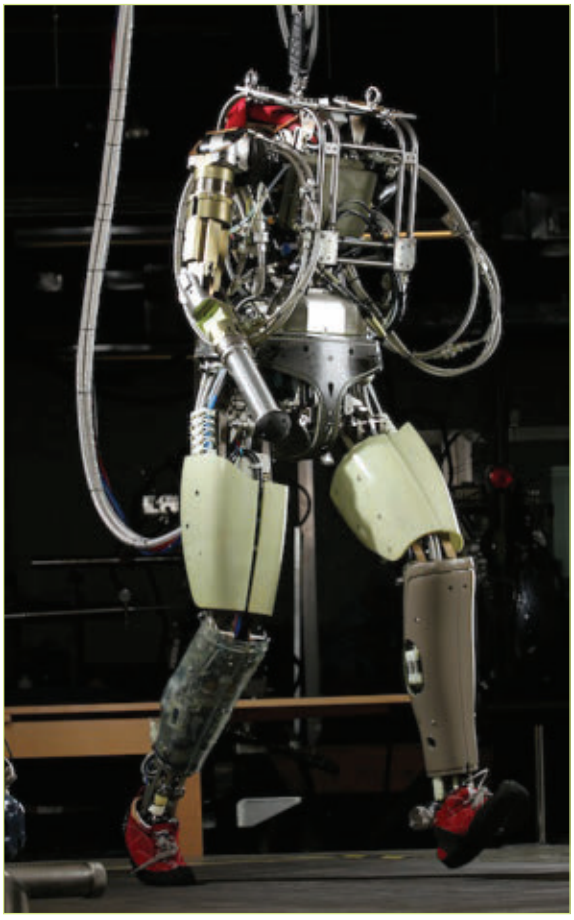
The next time you find yourself in a South Korean prison, this not especially friendly looking robot is going to be either your new best buddy or your new worst enemy.

It's huge and presumably very expensive, but aside from a microphone, a camera, some flashing lights, an alarm, and what looks to be an off-the-shelf Kinect sensor, it doesn't seem to really be able to do much.

The sophisticated part might be the software, which looks to be able to analyze behavior and make decisions as to when to alert a human operator that something is up.

Apparently, the next step is a robot that "conducts body searches" which could be particularly unpleasant for the end user. But that might be the whole idea. How likely are you to try and sneak contraband into a prison if you encounter a robot snapping a latex glove over its steely, probe-like fingers?

Photo by Jason Dorfman, CSAIL/MIT.



EVERYTHING HUMANOIDLY POSSIBLE

A recent notice posted on the US Federal Business Opportunities website confirms that DARPA has selected Boston Dynamics as a "sole source" to develop and build the humanoid robots that the software teams will use in the DARPA Robotics Challenge. "Of the few existing humanoid robots," the notice says, "[Boston Dynamics] was deemed to be the sole viable supplier for providing the necessary robotic platform capability within the specified timeframe." Boston Dynamics will build eight identical humanoids which will be based on PETMAN and Atlas — robots that the firm built for the US Army and DARPA, respectively. The result is expected to be "a one of a kind humanoid robot with state-of-the-art capability."

The final form of the humanoid is expected to have two arms, two legs, a torso, and a head, and will be physically capable of performing all of the tasks required for the disaster response scenarios scheduled in the Challenge." These tasks include driving cars, walking across rubble, moving heavy objects, opening and closing doors and valves, using tools, and climbing ladders.

One of the primary differences between the final platform and the current platform is going to be the addition of a head, which will likely contain the sensors allowing for the hybrid autonomous and teleoperated control that DARPA seems to have in mind for the Challenge (stereo cameras and LIDAR). Otherwise, what you'll expect to see is a 150 kg humanoid with a pair of two or three fingered hands.

Boston Dynamics is very well known for developing "dynamic" robots — machines that rather than using static techniques to control their motion move dynamically, relying on advanced control software and high performance hydraulic actuators. The trade-off is that hydraulics generally demand a fairly beastly engine.

"We're not going to disallow tethers in order to power the robot, but the tether will not be able to go all the way back to the operator. So, what the utility vehicle can help with is you can put the power supply for the robot on the flatbed of the vehicle and then the tether can go to the robot, and the vehicle is now a movable base that the robot can operate a certain distance from," commented program manager Dr. Gill Pratt.

There's nothing that says teams have to use a humanoid platform at all, and DARPA has been very specific that the Challenge is "decidedly not exclusive to humanoid systems." There are lots of reasons why humanoids are a good idea, but that in no way means that humanoids are necessarily the best idea. Of course, completing tasks involving equipment that's been built for humans (like climbing a ladder) would be quite difficult for non-humanoid robots, unless some creative solutions are implemented.

So, what about other humanoid robots? Well, there are some possible contenders, including ASH/SAFFiR (funded by the US Navy), as well as Meka and UT Austin's hyper agile biped.

The Challenge will last 15 months from October 1, 2012 to December 31, 2013, at which point there will be a competition. A second phase will happen from January 1, 2014 to December 31, 2014, ending with another competition where teams will try to improve their performances.

HERB REACHES FINAL FRONTIER

Herb — the Home Exploring Robot Butler — has been hard at work at Carnegie Mellon's Robotics Institute learning how to be —you guessed it — a home exploring robot butler. Siddhartha Srinivasa's group has effectively ended robotics research as we know it by teaching Herb to microwave frozen food. Yep, that's it. No more funding, no more papers, no more conferences. Robots can now do everything we could ever want.

For the upcoming Challenge itself, DARPA says robots will compete with each other, performing disaster response operations in a scenario that will likely include the following sequence of events:

1. Drive a utility vehicle at the site.
The robot has to enter the vehicle, drive it on a travel course, and exit the vehicle. The robot has to operate the vehicle controls including steering, throttle, brakes, and ignition. The vehicle is expected to be an unmodified utility vehicle such as a John Deere Gator or Polaris Ranger.

2. Travel dismounted across rubble.
The robot has to cross terrain ranging from smooth and level to rough and sloped, with some loose soil and rocks. A human would easily traverse the terrain. In addition, the terrain will include discrete obstacles such as rocks, bushes, trees, and ditches that the robot must avoid.

3. Remove debris blocking an entryway.
The robot has to move an object blocking an entryway. The object will have size, weight, and other properties to be movable either by a person or by the GFE (Government Furnished Equipment) Platform. The object is expected not to exceed five kilograms and be solid like a rock or a cinder block, and may have an irregular shape.

4. Open a door and enter a building.
The robot has to operate a door handle and have the strength to push the door open. The door and door handle are expected to be standard commercially available items.

5. Climb an industrial ladder and traverse an industrial walkway.
The robot has to traverse an industrial elevated walkway (also known as a catwalk) with a grated surface and handrails. As part of this task, the robot has to climb an industrial ladder. It is expected that a person would need to use both arms and legs to climb the ladder.

6. Use a tool to break through a concrete panel.
The robot has to use a power tool to perform "forceful manipulation." The power tool will likely be an air or electric impact hammer and chisel, or an electric reciprocating saw. The task is to break through a concrete panel (with no rebar) or through a framed wall.

7. Locate and close a valve near a leaking pipe.
The robot has to find a leaking pipe and a nearby valve, which it needs to close. The facility will contain multiple pipes, but only one will be leaking that will have visible smoke and an audible hiss similar to escaping gas. It is expected that a person would need to use two hands to close the valve.

8. Replace a component such as a cooling pump.
The robot has to locate the pump and be able to loosen one or more fasteners to extract the pump from its fittings, then reverse all the steps to replace the pump. It is expected that the pump will be small and compact enough that a human could handle it with a single hand.

DARPA says these are representative tasks and it "will adjust the difficulty of the scenario as the program progresses, depending on capabilities demonstrated and practical considerations." The scoring criteria and competition rules have not yet been defined. However, DARPA says that robots that perform more autonomously and consume less energy will score higher.

AFRON NATION

Roboticists in Africa and the United States have launched an initiative to enhance robotics education, research, and industry in Africa. The African Robotics Network (AFRON) wants to mobilize a community of institutions and individuals working on robotics-related areas, strengthening communication and collaboration among them.

"There are many robotics activities emerging in Africa," says Ayorkor Korsah, a professor of computer science at Ashesi University, in Berekuso — a 45 minute drive north of Accra, Ghana's capital. "Our goal is to highlight, enhance, and provide support for efforts in different parts of the continent."

Korsah co-founded AFRON with Ken Goldberg, an IEEE Fellow and professor of robotics at the University of California, Berkeley. Goldberg, who was born in Nigeria where his parents were teachers, says one of the first projects AFRON is planning involves an international competition to design an extremely low cost programmable robot for education. The idea (still under development) is to create a simple robot with parts costing under $10 that students would use to explore science and engineering topics. The robot would be connected via USB to a computer, and students would use open source software to program the robot's behavior and share their results. Goldberg acknowledges that developing a capable robot for just $10 is a challenge. "We want to get people thinking creatively," he says. "We are not sure it's possible, but it's a target to aim for." If they're successful, robots could become a powerful educational tool in Africa.

"Robotics is a great educational tool because it combines the tangible world with which kids are familiar, and the formalization of programming and mathematics," says Paulo Blikstein, a professor at Stanford's School of Education who studies the use of technology in classrooms, but is not involved with AFRON. "So, you get the best of both worlds, if it is done right."

Korsah (center, with her students in the photo) grew up in Nigeria and Ghana, and went on to study in the US, earning a Bachelor's and Master's degree from Dartmouth College and a Ph.D. in robotics and artificial intelligence from Carnegie Mellon University. Ashesi is a university founded by Patrick Awuah, a former Microsoft executive who returned to his native Ghana to establish a liberal arts college that he hoped would educate a new generation of African leaders.

Korsah says AFRON was inspired by other robotics initiatives such as the European Robotics Network (EURON), but while most networks have concentrated on research activities, AFRON focuses more broadly on education, research, and industry, including efforts aimed at exposing school children to robotics. Research labs, non-profit organizations, companies, and individuals may join AFRON for free; there are no dues. AFRON already has members from South Africa and Nigeria, and affiliated membership is open to anyone worldwide.

Images courtesy of Ayorkor Korsah/Ashesi University.

Goldberg, a member of IEEE Spectrum's editorial advisory board, says the plan is to organize projects, meetings, and events in Africa and at robotics and automation conferences abroad. He adds that — for the purposes of AFRON — "robotics" is broadly defined to include related areas such as automation, computer vision, signal processing, machine learning, mobile games, and other topics.

"The idea is to build bridges, connect people with common interests, and identify resources that can be shared," he continued. "We've already made progress within Ghana and are excited to reach across Africa and to include members in other countries."

GUNNING FOR IT

Local police recently used a robot with sonar to search an Oakland, CA estuary and found the .45-caliber semi-automatic handgun used in the Oikos University shooting. Oakland borrowed it from the San Francisco police department who also supplied a boat with underwater tracking. A tip was given by an expelled former student who admitted to the crime and had apparently targeted Wonja Kim — a former nursing program assistant director.

GONE SOFT

The holy grail of the whole soft robotics initiative that many research groups are so interested in seems to be the octopus. Anyone who has ever seen an octopus in action can certainly understand why. They're capable of some extraordinary maneuvers, thanks to relatively large brains, very fine motor control, and a near-total lack of bones. The Octopus Project is a European collaboration that's working on "investigating and understanding the principles that give rise to the octopus sensory-motor capabilities and incorporating them in new design approaches." Their newest design approach is this fully mobile roboctopus with eight soft tentacles.

Those big tentacles at the front are called "SMA arms" which means they're actuated by a shape-memory alloy that changes its length when heated, so no servos or anything is necessary. The other six arms are silicone with a steel cable inside, and this steel cable is attached to a bunch of nylon cables. By manipulating those nylon cables, the tentacle can be made to wiggle around and even grip things.

The ultimate plan is to have no rigid structures, so it can be an entirely soft robot with eight flexible arms, able to reach impracticable places, and simultaneously showing manipulation capability which could open up new scenarios for marine exploration and underwater rescue.

THE MIGHTY QIN

More details are emerging about BYD's upcoming car model, "Qin." Named after China's first empire, the vehicle is shaping up to be quite groundbreaking on a number of fronts.

The "Qin" plug-in — like its poor-selling cousin, the F3DM — uses BYD's own LiFePo4 battery. The LiFePo4 battery has a high energy density, can withstand up to 4,000 charges while still retaining 80% performance, and is environmentally benign as it uses no toxic heavy metals in its manufacture.

In the Qin, BYD opted for a smaller battery of 10 kwh instead of the 16 kwh model used on the F3DM. Due to its smaller size and improved design, the new battery is about 50% smaller and lighter than the one used on the F3DM. This reduced size also translates into reduced price, but does come at the expense of a slightly reduced all-electric range. The Qin is said to be able to go 50 km (31 miles) in all electric mode before the engine kicks in.

Besides more powerful electric motors (110 kw), the Qin dumps its 1.0L engine in exchange for BYD's own 1.5L turbo-charged direct-injected engine. This allows the car to reach a rumored 185 kph (115 mph) and accelerate 1-100 kph (0-60 mph) in just 6.9 seconds — a full two seconds faster than the Chevy Volt. Due to its smaller battery and increased wheelbase, the Qin will have more interior space than its predecessor.

The Qin has greatly improved styling over BYD's earlier models, most notably in the car's interior which is impressive. The car features two large TFT LCD displays and a mysterious "robot" on the dashboard. Details are sketchy at this time, but this robot appears to pop out from underneath the dashboard and is outfitted with cameras. It is likely the terminal of BYD's "i" networking system which handles everything from wireless Internet, Cloud computing, location services, music downloads, driver fatigue detection, and voice control.

Ultimately, the success or failure of the Qin will depend on product quality and price. With its recent warranty extension, BYD has leaped ahead of most of its Chinese rivals in the quality department. One thing is fairly certain — the Qin will not be cheap. The price of the vehicle is expected to come in around $24,000 to $32,000 USD.

Continued on page 72

Cool tidbits herein provided by Evan Ackerman at www.botjunkie.com, www.robotsnob.com, www.plasticpals.com, and other places.

COMBAT ZONE

Discuss this section in the *SERVO Magazine* forums at http://forum.servomagazine.com

Featured This Month:

BUILD REPORT:

Siafu and the Army of Ants – Part 1

● by Pete Smith

I've built a few Antweights (1 lb) over the years. It's never been a weight class I was particularly interested in, but with many events only hosting Ants and Beetleweights (3 lb) I have built them just to make up numbers or for my (then young) son to compete with.

However, I have perhaps the singular honor of never having won a fight with one! They all performed like the ill thought-out and hastily-built machines they were. Their wheels would fall off and the speed controllers would burn out or would simply decide not to move at all once they were in the arena.

With the success that the Weta and Trilobite Beetleweight kits are having through Kitbots (**www.kitbots.com**), I decided it was time to design and build a really competitive Antweight. I named it Siafu — the Swahili name for the fearsome Army ants of the genus Dorylus.

I decided my first serious Ant would be a drum bot. I've watched the success of Team Pneusance's Poco Tambor and decided I wanted to build a bot as (or more) effective as that, but in a kit form that would allow others to enter the sport with an effective bot right from the start.

I design my bots in 3D using SolidWorks and keep close track of the weight of all the components in a spreadsheet. The latter is very important because it's too easy to find yourself with a bot that is overweight for a competition, or one that has an ineffective weapon or inadequate armor.

Fingertech Robotics (**www.fingertechrobotics.com**) is the leading supplier for Antweight parts, and their TinyESC v2 speed controllers and Silver Spark drive motors are becoming the standards by which all others are measured. With

that in mind, I based my first layout around them.

Figure 1 shows two Silver Sparks, Lite Flite wheels, and a 36 mm brushless outrunner.

The 36 mm outrunner was chosen because it has a 5 mm shaft. I envisioned replacing it with a new shaft that would run the full width of the bot, and placing the outrunner within the drum itself. It did not take long, however, to find problems with the design. The Silver Sparks are quite long and this forced the bot (and hence the drum) to be quite wide. Similarly, the large diameter of the outrunner forced the drum to be even larger. A few trial chassis and drum designs showed that keeping such a design within 16 ounces would be very difficult. The long shaft on the outrunner would also make replacing the drum very difficult if it was damaged.

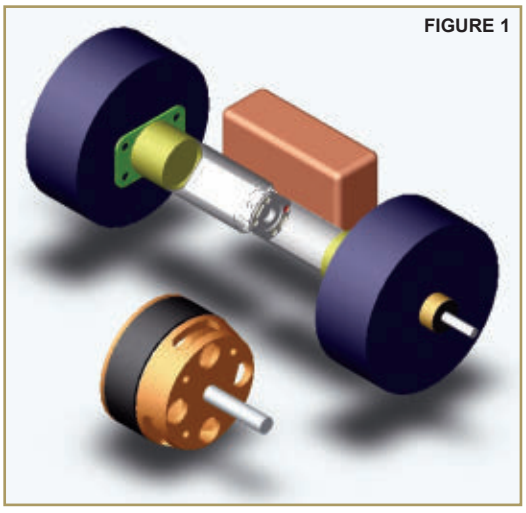


FIGURE 1

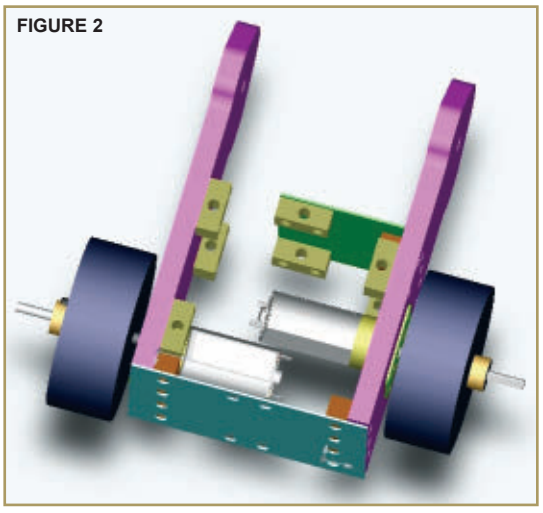


FIGURE 2

I then tried a design (**Figure 2**) that staggered the Silver Sparks and would use a more conventional inrunner and belt drive for the drum (similar to that on Weta). This reduced the width and the weight of the bot. While the staggered wheels might have made driving a little more interesting, I felt this could be made to work. Unfortunately, a problem arose with the belt drive system. The inrunner took up a lot of space, and belts and pulleys that small would be problematic to attach to the drum. Again, I had run into a dead end.

The answer came when I found a small motor (**Figure 3**) at the same supplier as I use for the Kitbots 1,000 RPM Beetle motors. These motors were listed as 1,000 RPM at 6V and used a motor about one half the length of that on the Silver Sparks. The question to be answered was, would these smaller motors have enough power to drive an Ant?

To answer that, I built a small test chassis (**Figure 4**). It used two of the new motors, a couple of the TinyESC speed controllers, a 2S LiPo battery, and 1.375" Banebots wheels. It was then weighted up to close to one pound and given some extensive testing.

The motors proved to have more than enough speed, and did not heat up even after three minutes. However, they lacked pushing power. The same motor is also available with 500 RPM at 6V, and I decided that would be ideal. I increased wheel size a little and that resulted in a good balance of speed and power.

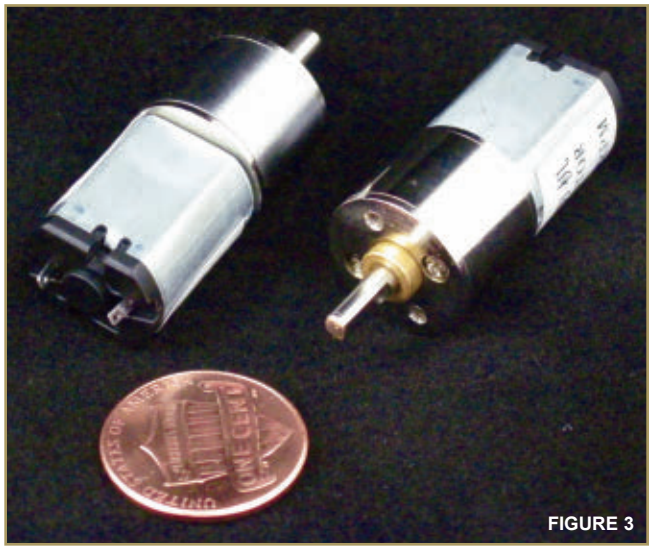


FIGURE 3

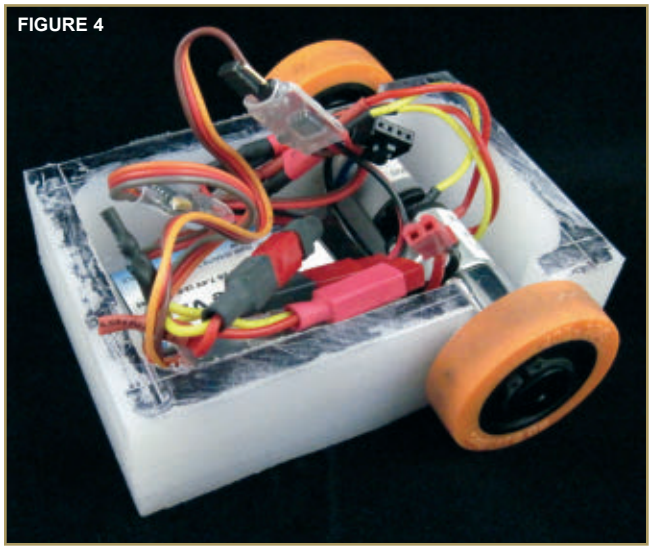


FIGURE 4

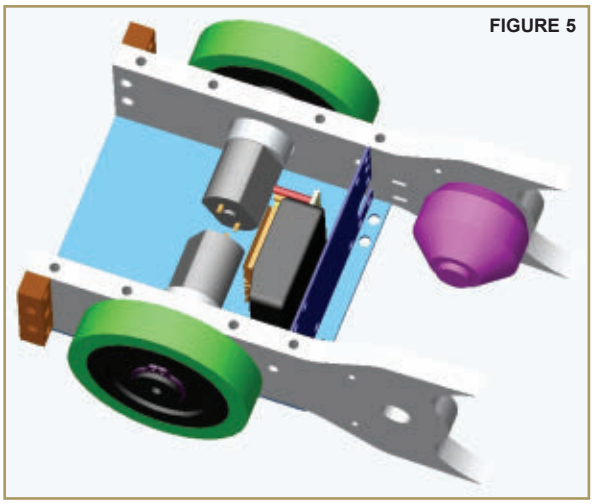


FIGURE 5

This left the problem of how to power the drum. Poco Tambor had been successful for years and its builder, Chuck Butler, kindly showed me how its drum worked. It uses the outrunners 1/8" shaft at one end of the drum and a 1/4" bolt running in a bronze bearing at the other. The 1/8" shaft allowed me to choose a smaller 28 mm diameter outrunner which could fit in a smaller drum. The motor chosen had 1,740 kV which would give a drum RPM of about 12.5K at 7.2V.

The final layout (**Figure 5**) shows the small drive motors, 1.875" Banebots wheels, and the new 28 mm outrunner. Getting the motor into the drum again frees up plenty of space for an 18A reversible speed controller for the weapon.

Part 2 of this series will complete the design process and provide a list of the parts used. **SV**

EVENT REPORT:

Happy 10th Birthday for Motorama – Motorama 2012

● by Pete Smith

This February 17th–19th, the Northeast Robotics Club (**www.nerc.us**) hosted the 10th annual combat event at the Motorama Motorsports Extravaganza and Custom Car Show (**www.motoramaevents.com**) in Harrisburg, PA.

One hundred robots were entered in a total of seven weight classes from the 250g Fairyweights to the 30 lb Featherweights. Ninety five passed through safety checks and actually competed. This figure was down only slightly from last year, so the sport seems to be weathering the recession reasonably well.

As usual, the two smallest weight classes did battle on the Friday before the main show opened. These weight classes are fun to compete in, but lack the size and action required for a paying audience in a venue on the scale of the size provided by Motorama. The fights for these weight classes are fought in a small 8 x 8 area (which doubles as a very useful test box for the big bots during the main event).

Nine 250g Fairyweights and 20 1 lb Antweights took part.

While this competition took place, NERC members and volunteers assembled the main 16 x 16 arena. The weight limits for robots are decided by the arenas. The big arena's 1/2" of polycarbonate plastic "glass" is only really safe for bots up to the 30 lb Featherweight class.

In the Fairyweights, Kongol with its moving blade arm had a good run while a new bot, Tracked Terror, overcame drumbot Lolcat and then the wedge Baby V to get first place.

The Antweights were hard fought. Wedges Antelope and Kobalos, and full body spinner See You Next Wednesday (**Figure 1**) and bar spinner Vile Ant made it through to the semis, and Vile Ant

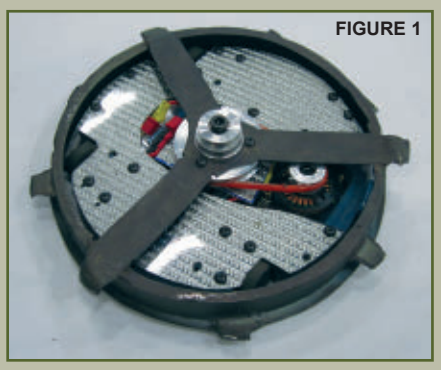


FIGURE 1

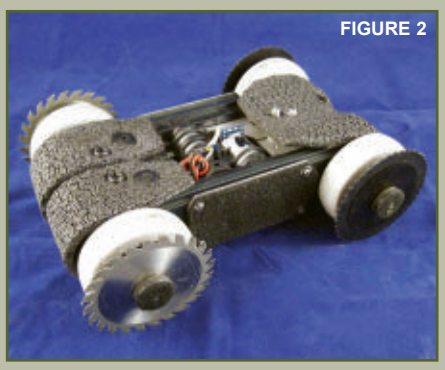


FIGURE 2

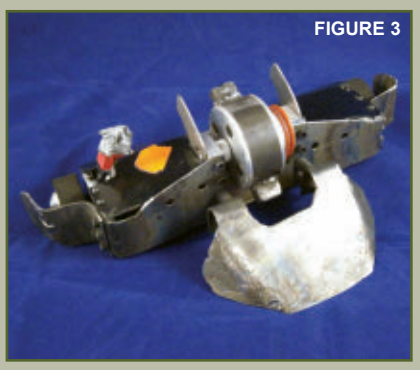


FIGURE 3

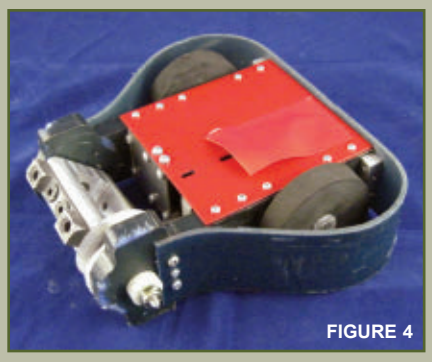


FIGURE 4

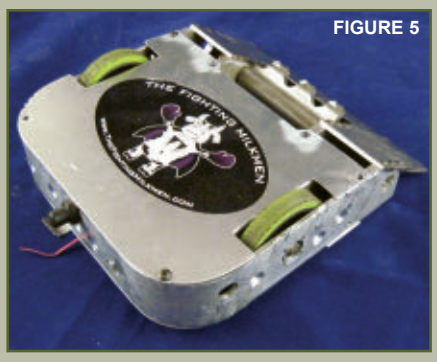


FIGURE 5

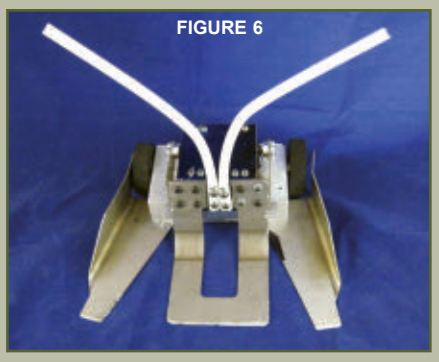


FIGURE 6

only won in the final after getting through Kobalos' rear armor and removing both his wheels.

The main event for the Beetleweights and heavier bots got underway on Saturday.

The fights started around noon and continued with only a short break for lunch until about 6.00 pm. They restarted the next morning with the last finals held around 4 pm on Sunday.

Twenty one bots fought it out in the 3 lb Beetleweights. A heavily modified RC toy Chobham (**Figure 2**) had a good run. Strong competitor Mr. Croup (**Figure 3**) struggled with reliability as did Weta, God of Ugly Things (**Figure 4**) after its tough match against Skim (**Figure 5**).

The dark horse of the weight class was long-time competitor The Revenge of Doctor Super Brain (**Figure 6**). This year, it was back with a titanium body and lifter mechanism. It survived a loss in the first round to Traumatizer (**Figure 7**), then fought its way back up through the losers brackets to dominate super tough wedge Shame Spiral in two finals to take first place.

Last year's 12 lb Hobbyweights Champion Cataclysm was not entered this year but a couple of new drumbots, Upchuck (**Figure 8**) and Wardrums, and seasoned veterans Zandor, Fiasco, and Apollyon (**Figure 9**) helped make up a strong field.

Upchuck looked powerful until it met Fiasco. A single blow from Fiasco's blade revealed Upchuck's Achilles heel. Its top and bottom panels were only retained by a few screws and it was these panels that maintained the squareness and integrity of the chassis. The impact placed so much strain on the screws that they failed, and this allowed the chassis to "lozenge" with spectacular results (**Figure 10**).

Apollyon had a straight run through the winners brackets before meeting and beating Zandor in the finals.

The 15 lb BotsIQ high school class was dominated by the very effective drum bot Pixie (**Figure 11**) which overcame all opposition to win by beating KV in the finals.

The Sportsman class was dominated again by pneumatic

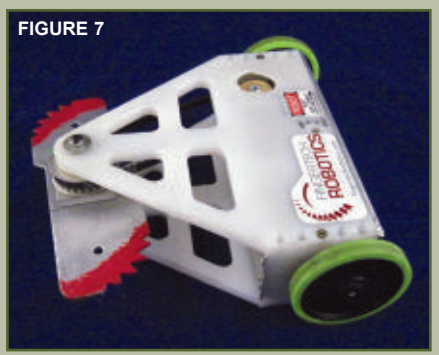


FIGURE 7

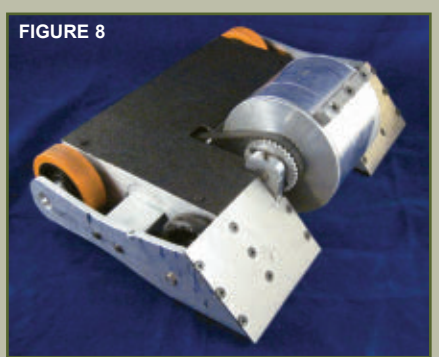


FIGURE 8


FIGURE 9


FIGURE 12


FIGURE 11

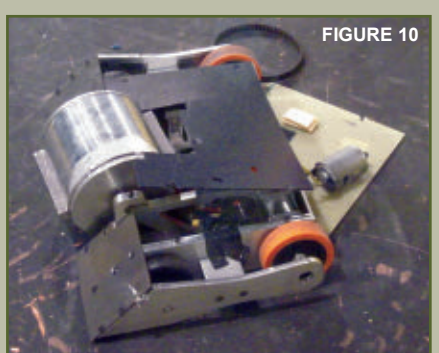


FIGURE 10

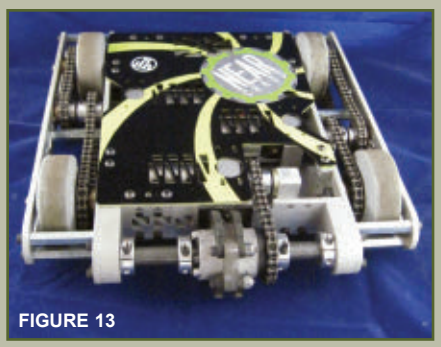


FIGURE 13

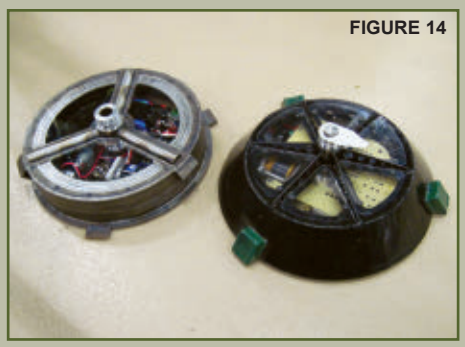


FIGURE 14

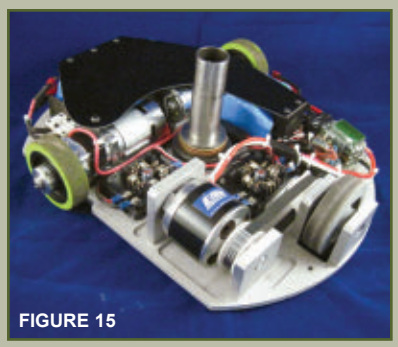


FIGURE 15


FIGURE 16

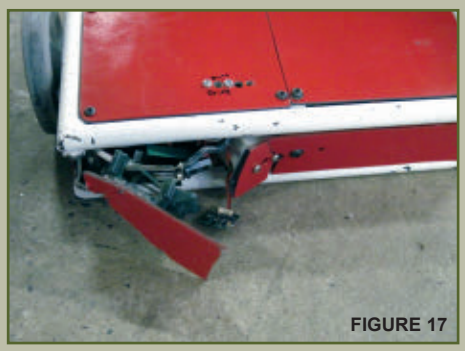


FIGURE 17

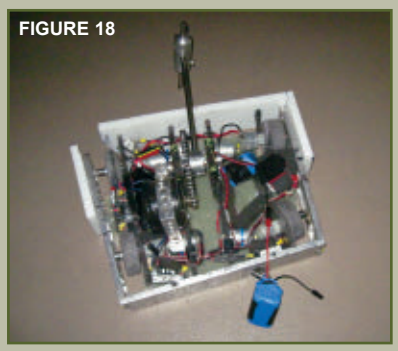


FIGURE 18

flippers. There were two this year: Upheaval and Phoenix. The former again combined excellent performance and good driving to take first place. One non-flipper — the crusher Lockjaw (**Figure 12**) — gave Upheaval a good run, only losing narrowly to it in the final. A new bot Nyx (**Figure 13**) showed promise and is likely to be a force in future events.

The most powerful bots that fight at Motorama are the 30 lb Featherweights. This year, there was a very strong field including the fearsome bar spinners Sloth and Moros, the vertical spinners Shaka and Lenis Tomb, a tough steel wedge Pinball, and not one but two full body spinners (**Figure 14**; Tetanus) that took second place at Franklin in the fall, and Ill-Tempered, a newly completed and very impressive design (**Figure 15**).

There were some memorable fights. Shaka looked about to take Mangi apart when a lucky/skillful hit punctured Shaka's LiPo battery pack which resulted in a lot of smoke and excitement as the bot was quickly taken outside and blasted with a CO_2 extinguisher (**Figure 16**).

Ill-Tempered had weapon speed controller issues during the event, but demonstrated its ability by tearing open Sloth (**Figure 17**). Tetanus got revenge in spade-fulls against Mangi (**Figure 18**) for its loss in the finals at Franklin and went on to win its first major tournament.

The event finished on time and the prizes supplied by sponsors Dimension Engineering, Fingertech Robotics, Kitbots, Holmes Hobbies, and Castle Creations were awarded by a couple of the "Miss Motorama" contestants. Many of the competitors stayed on to help with dismantling and loading up the big and small arenas, plus all the other tidying up required in the pits area.

The traditional post-Motorama dinner was held at the Texas Roadhouse and a good time was had by all.

It's been a great 10 years at Motorama and we all hope that the next 10 years are as good or even better! SV

Results

	150g Fairyweights	1 lb Beetleweights	3 lb Beetleweights
1st:	Tracked Terror	Vile Ant	Revenge of Dr Super Brain
2nd:	Baby V	Kobalos	Shame Spiral
3rd:	Lolcat	See You Next Wednesday	Weta, God of Ugly Things

	12 lb Hobbyweights	30 lb Featherweights	30 lb Sportsmen
1st:	Apollyon	Tetanus	Upheaval
2nd:	Zandor	Lenin's Tomb	Lock-Jaw
3rd:	Tough Nut	Gloomy	Phoenix

Coolest Robot	Enforcer
Best Driver	Mike Jeffries (Kobalos, Apollyon)
Most Destructive	Tetanus
Kitbots "Next Step" kit	Team OpenRobotix (Chobham)

Photos by author. Many of the fights mentioned above can be found on YouTube by searching on the bot's name and "Motorama 2012."

EVENT REPORT:

Havoc at Harrisburg University – The 2nd Annual Downtown Dogfight

● by Dave Graham

Robotic enthusiasts with brainy and brawny bots converged on Harrisburg University in March to compete in multiple events at the 2nd Annual Downtown Dogfight (**Figure 1**). Hosted by Harrisburg University, the annual event is the collaborative effort of PennBots, the Robot Club of Pennsylvania, and Harrisburg University.

This year's venue included an Insect class fighting robot competition consisting of 150 gram Fleas (a.k.a., Fairies), one pound Ants, and three pound Beetles, as well as line following and maze solving challenges for thinking robots. The schedule alternated between fighting and thinking events, affording the fighters a chance to repair damage and the thinkers time to tweak their code for best bot performance.

I have to give a shout-out right up front to Harrisburg University and their state-of-the-art facilities. The space was huge, the pits were carpeted, and there were plenty of hexagon workstations sporting pop-up power receptacles. The place was full of creature comforts and is easily the nicest facility I've ever seen for a robot competition. What really puts the Harrisburg University facility over the top is the option for spectators to view the event from a second floor balcony, affording them a unique view from above the arena and of fighting robot matches (**Figure 2**).

The fighting competition started a little slowly when only three Fleas came forward to tow the line. The Fleas were all wedge bots, and the match record was tied at 1-1 for all three bots after the first round of the round-robin format. Heath Hill

FIGURE 1. Group shot of competitors.

FIGURE 2. View of the arena from balcony.

FIGURE 3. Fleaweight bot Baby V.

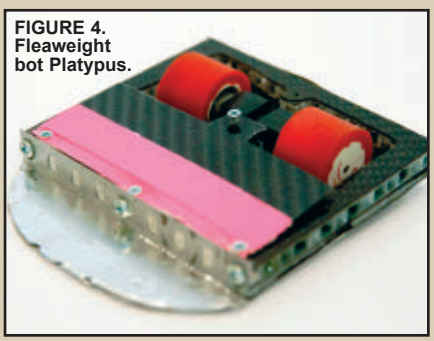


FIGURE 4. Fleaweight bot Platypus.

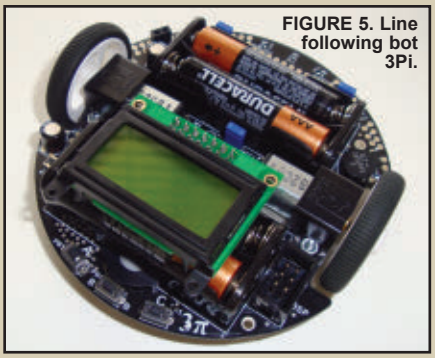


FIGURE 5. Line following bot 3Pi.

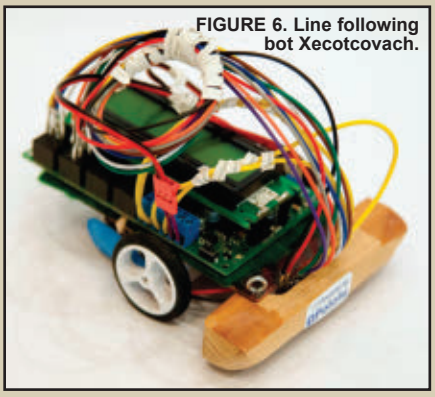


FIGURE 6. Line following bot Xecotcovach.

FIGURE 7. Lief Tvenstrup watching his line following bot Ascahil while Bill Bechtel looks on.

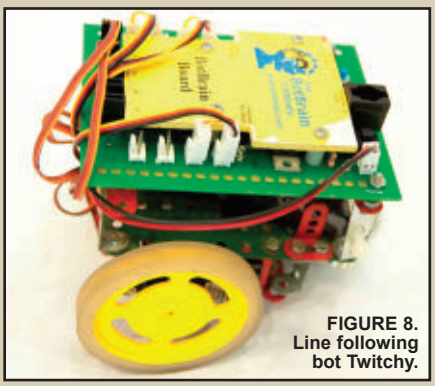


FIGURE 8. Line following bot Twitchy.

driving Baby V (**Figure 3**) took control in the second round going 2-0 and taking first place. Platypus (**Figure 4**) was my first attempt at using a Pololu Baby Orangutan controller as a speed controller and the new red "Cobra" mini Sumo tires from Fingertech Robotics. Overall, the bot ran well but could only manage a third place finish.

Line following was a new event at this year's competition and drew an impressive group of six robots designed around LEGO NXT, Pololu, PICAXE, Parallax, and BotBrain products. My stock Pololu 3Pi bot (**Figure 5**) smoked the closed-loop track, completing two laps in less than 16 seconds — almost twice as fast as the nearest competitor. Krishna Iyer's bot Xecotcovach (**Figure 6**) used a collection of Pololu products, including an Orangutan robot controller, sensor array, motors, and wheels. Xecotcovach was impressive, but repeatedly overshot the corners resulting in a second place finish. Competitor Lief Tvenstrup watched his LEGO NXT bot Ascahil navigate the track while scorer Bill Bechtel looked on (**Figure 7**). Bechtel's BotBrain creation Twitchy (**Figure 8**) took third place.

The Antweight competition got off to a fast start when Kyle Singer's spinner bot Szalor (**Figure 9**) connected with Glen Warner's Nuggenator, stopping Nuggenator dead in its tracks and forcing him to tap out. In second round action, my latest Ant creation Adobe (**Figure 10**) drew Szalor. It was a classic example of an equal but opposite reaction when Szalor's spinner ripped into Adobe's armor, and in the process propelled itself into the pit. Szalor's day would end in the loser's bracket when Richard Kelley's bot Little Box gave him a ride to the pit (**Figure 11**).

Alex's Udanis' Ant bot The Wired Wedge (**Figure 12**) deposited the always tough Ken Brandon's bot Buzz (**Figure 13**) in the pit. Also in second round action, Alex Udanis' other Ant bot PAID gave Heath Hill's bot Junkyard Dog a ride to the pit (**Figure 14**). Third round action was strange in that my two bots Adobe and Mateo faced one another (Mateo forfeited), and Alex Udanis' two bots PAID and The Wired Wedge faced one another (The Wired Wedge forfeited). PAID then beat Adobe to move on to the Ant final match.

In the Ant loser's bracket, Richard Kelley's bot Little Box took out Nuggenator and Szalor before going down to The Wired Wedge. Buzz did a little gyro dancing before launching the Junkyard Dog into the pit (**Figure 15**) and went nose-to-nose with Mateo (**Figure 16**) before sticking Mateo on the arena wall. Buzz eventually fell to The Wired Wedge.

In the Ant semi-final match, The Wired Wedge eliminated Adobe, resulting in a rematch of Alex Udanis' bots PAID and The Wired Wedge for the championship. The Wired Wedge forfeited giving PAID the Ant gold.

This year's maze solving competition was greatly improved from last year, featuring seven maze solvers built around LEGO NXT, PICAXE, Parallax, and BotBrain engines. The course was a random arrangement of a standard BotBrain Educational Robots Company maze kit. Competitors could start from

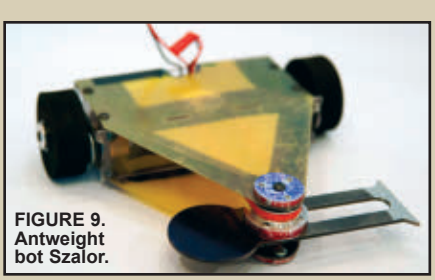


FIGURE 9. Antweight bot Szalor.

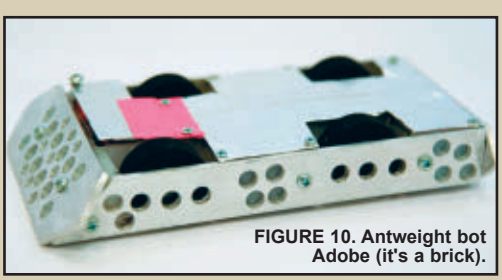


FIGURE 10. Antweight bot Adobe (it's a brick).

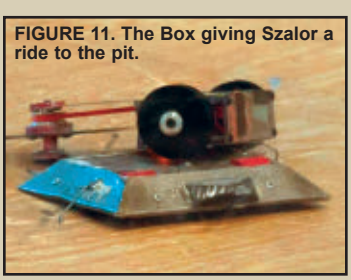


FIGURE 11. The Box giving Szalor a ride to the pit.

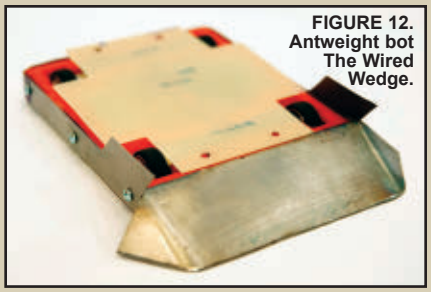


FIGURE 12. Antweight bot The Wired Wedge.

FIGURE 13. Antweight bot Buzz.

FIGURE 14. PAID giving Junkyard Dog a ride to the pit.

FIGURE 15. Buzz (right) gyro dancing for Junkyard Dog (left).

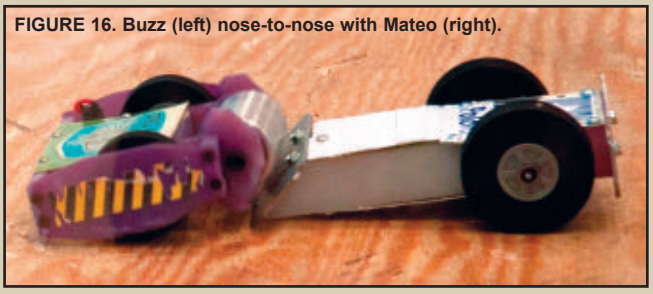


FIGURE 16. Buzz (left) nose-to-nose with Mateo (right).

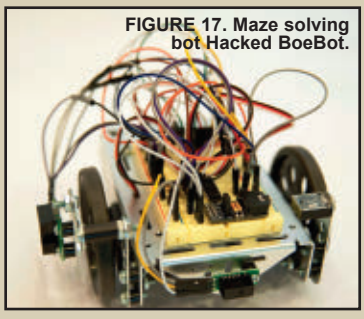


FIGURE 17. Maze solving bot Hacked BoeBot.

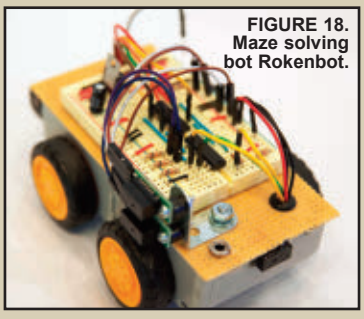


FIGURE 18. Maze solving bot Rokenbot.

FIGURE 19. Aaron Derk watching his bot AUTGO.

FIGURE 20. Justin Pennypacker launching his bot Sonic BOE-Bot.

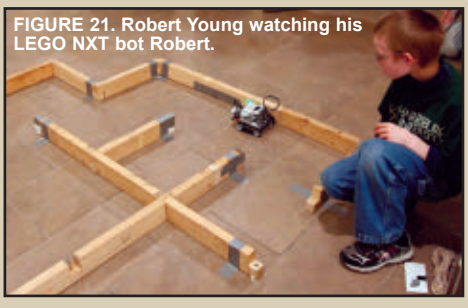


FIGURE 21. Robert Young watching his LEGO NXT bot Robert.

FIGURE 22. Play 'n Crazy rips the wheel from Prototype.

either end of the maze. My bot Hacked BoeBot (**Figure 17**) — designed around a PICAXE processor on a BoeBot chassis with Pololu sensors — won in a close competition. I also took second place with Rokenbot (**Figure 18**) using another PICAXE processor on a Rokenbok chassis with Pololu sensors. Competitor Aaron Derk watched his BotBrain creation AUTGO navigate the course to a third place finish (**Figure 19**). Teen competitor Justin Pennypacker launched his bot Sonic BOE-Bot to a fourth place finish (**Figure 20**).

Robert Young watched his LEGO NXT bot Robert use the right-hand rule to follow the wall (**Figure 21**).

The Beetleweight competition is best described as a destructive, crowd-pleasing lesson in blunt-force trauma. Brandon Young and his bot Play 'n Crazy started the action by ripping the wheel off Heath Hill's bot Prototype (**Figure 22**). Young chose to run his beater spinning downwards as opposed to the usual upwards, resulting in his bot repeatedly launching itself after making contact with an opponent.

In second round action, Topsy Turvey (**Figure 23**) sent Play 'n Crazy to the loser's bracket, Hunter Trout's bot Hammerhead won a pushing contest against Josh Lafferty's bot Spikes, and Richard Kelley's bot The Box gave Tim Thompson's bot SID a ride to the pit (**Figure 24**).

The upset of the day came when my new Beetleweight bot Sidewinder (**Figure 25**) faced Kyle Singer's bot Ripto (**Figure 26**). After a thunderous opening hit from Ripto rocketed Sidewinder off the top of the arena, Ripto came in for the kill. Sidewinder spun into Ripto's blade resulting in Ripto flipping itself up

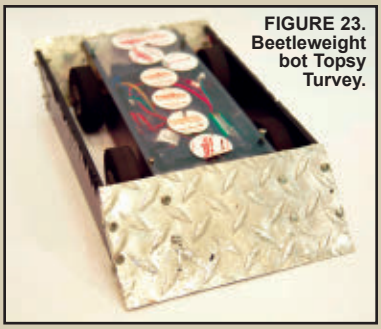


FIGURE 23. Beetleweight bot Topsy Turvey.

FIGURE 24. The Box giving SID a ride.

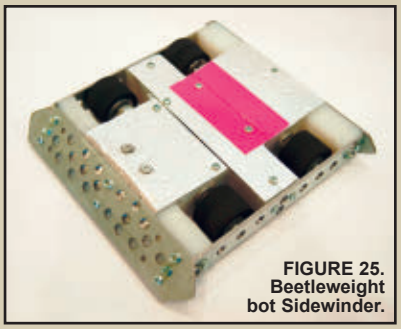


FIGURE 25. Beetleweight bot Sidewinder.

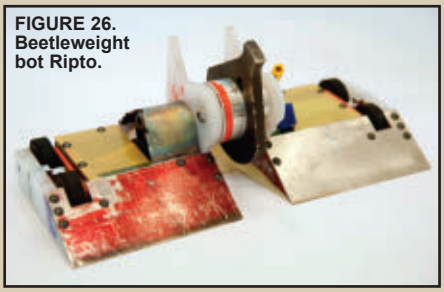


FIGURE 26. Beetleweight bot Ripto.

FIGURE 27. Ripto stuck on the arena wall.

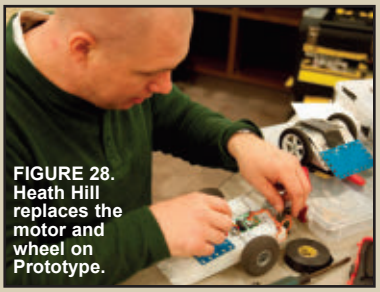


FIGURE 28. Heath Hill replaces the motor and wheel on Prototype.

FIGURE 29. Play 'n Crazy rips the wheel from Prototype.

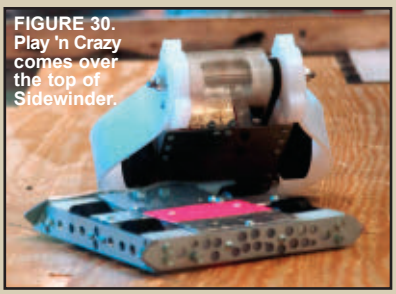


FIGURE 30. Play 'n Crazy comes over the top of Sidewinder.

FIGURE 31. Ripto launches Topsy Turvey.

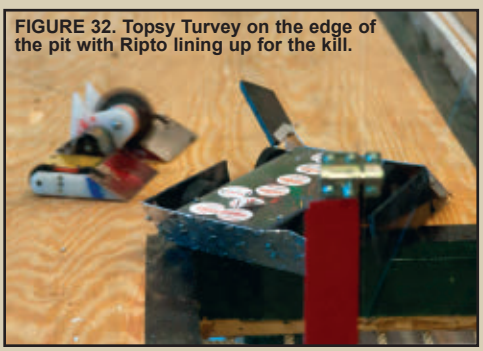


FIGURE 32. Topsy Turvey on the edge of the pit with Ripto lining up for the kill.

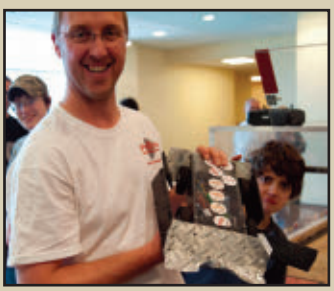


FIGURE 33. Bill Bechtel with his bot Topsy Turvey after the match with Ripto – son Max frowns from behind.

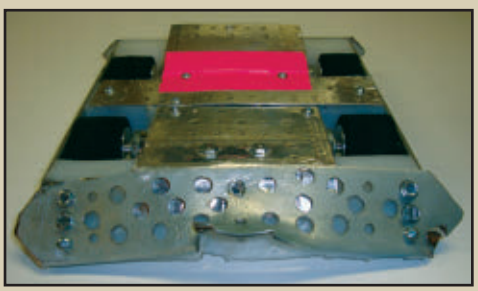


FIGURE 34. Sidewinder after second match with Ripto.

against the arena wall (**Figure 27**). Ripto was stuck and counted out for non-movement. Sidewinder was cosmetically repaired with a hammer rhinoplasty.

In third round action, Hammerhead and The Box pushed their opponents Topsy Turvey and Sidewinder, respectively, into the pit. The fourth round of the winner's bracket saw The Box defeat Hammerhead to move to the Beetle final match.

Heath Hill replaced the wheel and motor on his bot Prototype (**Figure 28**) and went on to defeat Spikes in the loser's bracket, only to be paired again with Brandon Young and Play 'n Crazy. It looked like a replay of the first match with Play 'n Crazy ripping the wheel and the motor out of Prototype and again launching itself in the process (**Figure 29**). Sidewinder would end Play 'n Crazy's day when Play 'n Crazy came over the top of Sidewinder (**Figure 30**) and drove into the pit.

The real story of the loser's bracket was Ripto. It appeared that Ripto was on a mission to launch its opponents into orbit. Ripto absolutely bludgeoned Topsy Turvey by splitting both of its side rails and chewing up the front plate before launching it skyward (**Figure 31**). When Topsy Turvey came to rest, it was on the edge of the pit (**Figure 32**) and Ripto gently pushed it into the pit to end the carnage.

Following the match, Bill Bechtel posed with Topsy Turvey while his

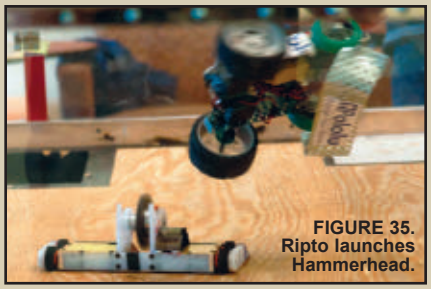


FIGURE 35. Ripto launches Hammerhead.

FIGURE 36. Ripto rips the wheel off Hammerhead.

son Max frowned from behind (**Figure 33**). Next was a rematch with Sidewinder. Ripto launched Sidewinder into the pit with one vicious hit. After the match, Sidewinder was looking forward to a second nose job (**Figure 34**).

Ripto made another attempt at putting an opponent into orbit by launching Hammerhead (**Figure 35**). After Hammerhead returned to earth, Ripto administered the final blow (**Figure 36**) and in the process inadvertently wrapped Hammerhead's dislocated tire around its spinner (**Figure 37**). Since the spinner was jammed, Ripto became a pusher and nudged Hammerhead into the pit for the win. After the match, Hunter Trout (left) looked down at Hammerhead while Kyle Singer (right) held Ripto (**Figure 38**).

Ripto's win sent it to the final match against Richard Kelley's bot The Box. Ripto needed to win two straight matches in the competition's double-elimination format to take top honors. In the first match, Ripto launched The Box (**Figure 39**) causing it to lose power and tap out. In the second match, the crowd was treated to a display of sparks as Ripto's spinner tried to eat into The Box's titanium armor. Ripto eventually pushed The Box into the pit for the win. Following the match, The Box and Ripto compared their battle scars (**Figure 40**).

A list of the fighting robot winners is in **Table 1** and a list of the thinking robot winners is in **Table 2**. The top three winners in each event received certificates and shared a bounty of merchandise

FIGURE 37. Ripto with Hammerhead's tire wrapped around its spinner.

FIGURE 38. Hunter Trout (left) and his bot Hammerhead, and Kyle Singer (right) and his bot Ripto after the match.

FIGURE 39. Ripto launches The Box.

FIGURE 40. The Box and Ripto compare battle scars.

TABLE 1 - FIGHTING ROBOT WINNERS

	FLEA	ANT	BEETLE
1st:	Baby V *Heath Hill*	PAID *Alex Udanis*	Ripto *Kyle Singer*
2nd:	The Baby Box *Richard Kelley*	The Wired Wedge *Alex Udanis*	The Box *Richard Kelley*
3rd:	Platypus *Dave Graham*	Adobe *Dave Graham*	Hammerhead *Hunter Trout*
Melee:	Platypus *Dave Graham*	The Wired Wedge *Alex Udanis*	Hammerhead *Hunter Trout*

TABLE 2 - THINKING ROBOT WINNERS

	LINE FOLLOWING	MAZE SOLVING
1st:	3Pi *Dave Graham*	Hacked BoeBot *Dave Graham*
2nd:	*Xecotcovach* *Krishna Iyer*	Rokenbot *Dave Graham*
3rd:	Twitchy *Bill Bechtel*	AUTGO *Aaron Derk*

from our very generous event sponsors FingerTech Robotics (**www.fingertechrobotics.com**), Pololu (**www.pololu.com**), Parallax (**www.parallax.com**), BotBrain Educational **Robots** (**www.botbrain.com**), and *SERVO Magazine* (**www.servomagazine.com**).

Mark your calendars now and plan to spend a Saturday in late March 2013 at Harrisburg University either watching — or better still — competing in the 3rd Annual Downtown Dogfight. You can follow PennBots on their website at **www.pennbots.org**, and Harrisburg University on their website at **www.harrisburgu.net**. SV

EVENT REPORT:

Gulf Coast Robot Sports Completes its 10th Event

● by Andrea Suarez

January 2012 marked the 10th South Florida Insectweight combat event from Gulf Coast Robot Sports (GCRS). Since March 2009, veteran robot builder Jim Smentowski has been hosting these events every 2-3 months at The Robot MarketPlace store in Bradenton, FL. They usually average 15-25 bots ranging from 150 grams to three pounds. Robot builder and announcer extraordinaire, Seth Carr, has set up a live webcast of recent events so audiences from around the world — even friends from Puerto Rico and Canada — can sit at home and root for the South FL builders via comments on the feed as they settle the score in the 8 x 8 arena.

In the summer of 2010, GCRS 5 marked the first Insectweight event for our team — Busted Nuts Robotics — after years of building 120 lb bots. After several weeks of building our new robots, we loaded up the car and started the 3.5 hour trek from Miami to Bradenton. We pulled into The Robot Marketplace and were instantly welcomed by the familiar enthusiasm for new competitors and new bots that is so common in this sport.

Eleven year old Reid Kauffman joined the GCRS group for his first robotics event at GCRS 9. He brought his own Antweight robot — a modified RC car with a custom Lexan wedge body — and competed as his friends cheered him on. He was eager to learn as much as he could from all the competitors and he came back with an added ice pick weapon for GCRS 10. He even challenged a Beetleweight, DejaVoodoo, to a grudge match.

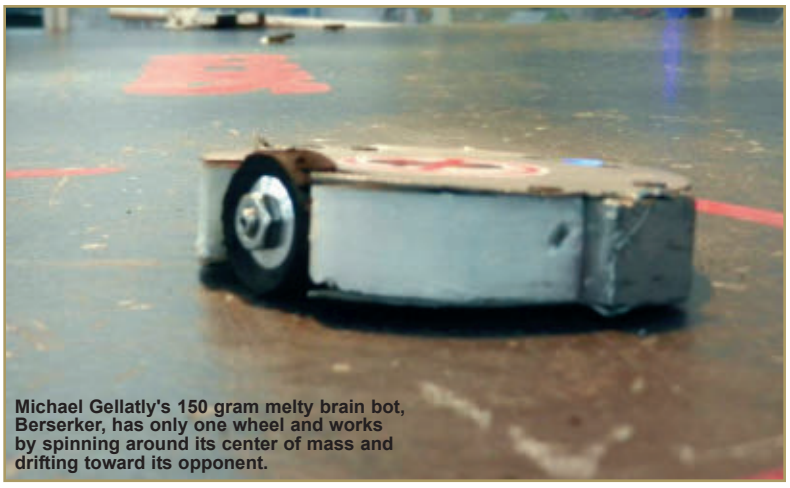
A new division for fighting unmodified RC cars lets anyone join in on the fun, and I brought my RC forklift, Crushy, to battle it out. My competitors ranged from toddlers controlling RC cars to store employees showing off their RC helicopter skills. Since then, Jim has added different events to keep the crowd engaged, including head-to-head robot driving challenges and a hex-bug round-up, all with different prizes.

There is no lack of exciting match-ups at GCRS. Dan Curhan's Antweight, Cannibal, shot his dad's robot, Thorn, into the ceiling, and it bounced off the floor with an impressive amount of energy. He shattered some of the ceiling lights when he sent my robot,

Getting ready to start at Gulf Coast Robot Sports 8.

Image by Dan Curhan.

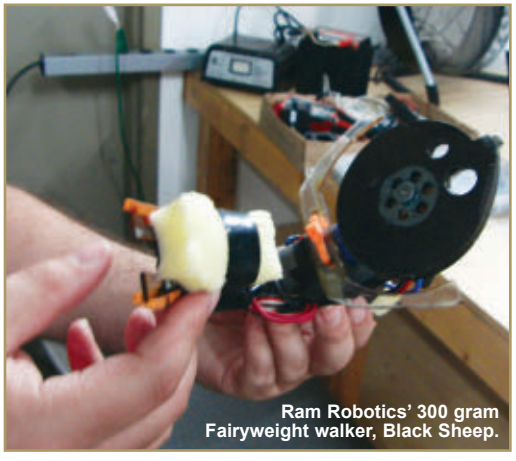
Reid competes his Antweight, BYE, as his friends watch the match.



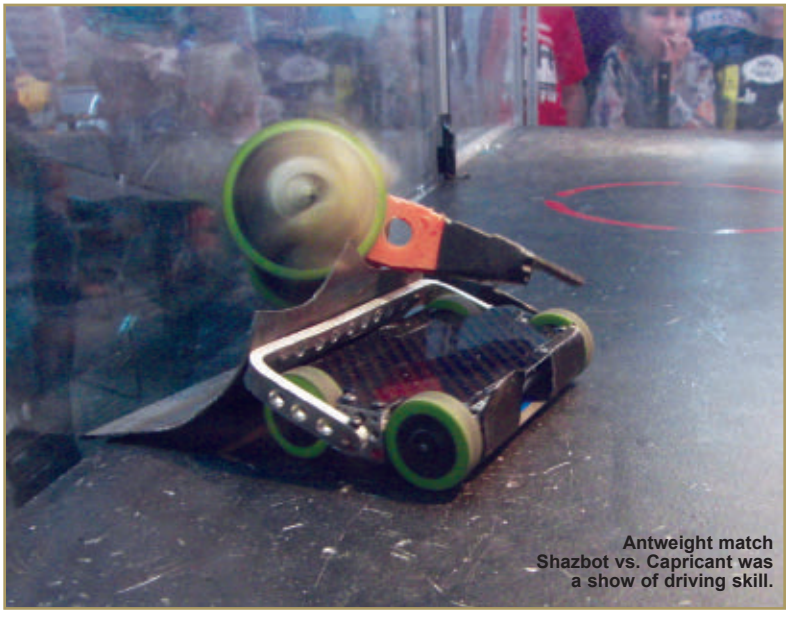
Michael Gellatly's 150 gram melty brain bot, Berserker, has only one wheel and works by spinning around its center of mass and drifting toward its opponent.

Fairyweight Nano Nightmare loses its weapon vs. Dirty Sanchez at GCRS 8.

MPX, Voodoo Magic, Walla Walla, and RamVac compete in the Beetleweight rumble at GCRS 5.



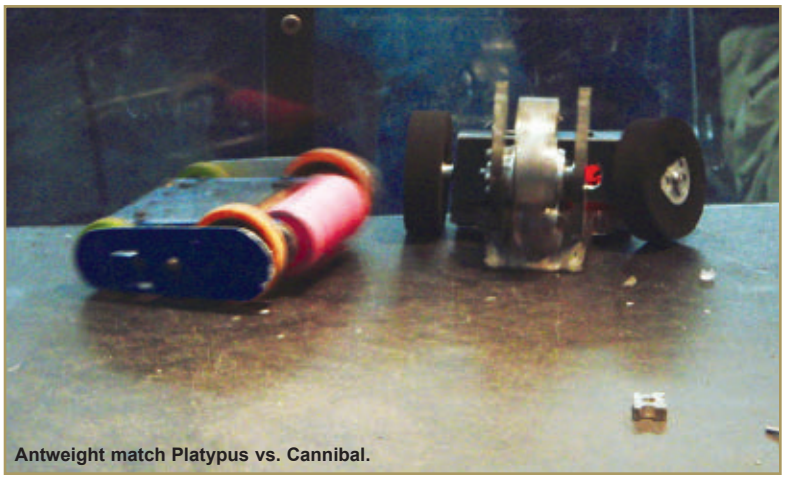
Ram Robotics' 300 gram Fairyweight walker, Black Sheep.

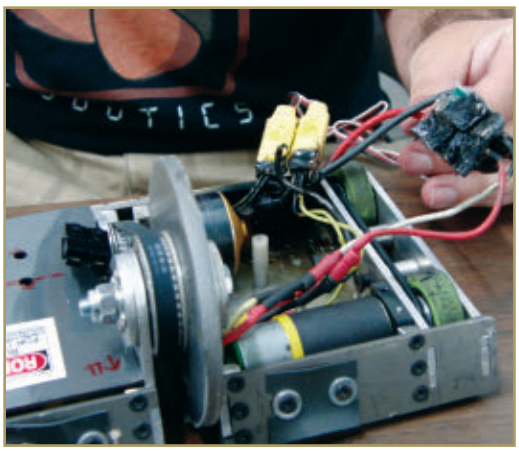


Antweight match Shazbot vs. Capricant was a show of driving skill.

Ram Robotics worked with Boy Scout Troop 415 to help them get their merit badge.

Celebrating Seth's birthday, robot style!


Antweight match Platypus vs. Cannibal.


Beetleweight Voodoo Magic caught fire during the final match against Leftie at GCRS 7.

WhipperSnapper, flying. Jim Smentowski always puts on a great show with his driving skills and quirky bots like Shazbot, an Antweight version of his 220 lb BreakerBox, and Nano Nightmare, a Fairyweight incarnation of his 220 lb Nightmare. Biggest Fairyweight bot has to go to Paul Grata's Pissed Off Unicorn, however, with its 8" spinning shell. Seth Carr's Antweight, Platypus, is now retired but did well at past events. He even lost his drum a few times as it bounced off the arena walls over and over again.

It is always exciting to see innovative bots push the sport forward. A few Melty Brain robots (translational drifting bots) have competed at GCRS. These robots spin their entire mass as they "drift" forward by pulsing one motor faster once every revolution to achieve translation. Richard Wong's brushless Antweight and Beetleweight melty bots have an incredible amount of energy, but each was eliminated after ricocheting three or four times off the wall and falling out the drop-out zone. Michael Gellatly from Busted Nuts Robotics won a few events with his one-wheeled Fairyweight melty, Berserker. Other innovative bots include Paul Grata's Beetleweight (also from Busted Nuts), Walla Walla, which uses the overhead spinning bar connected to an impeller to draw a vacuum and increase his robot's effective weight. Sam McAmis (Ram Robotics from New Port Richey, FL) built a 150g walker with a vertical disk to take advantage of the 100% weight bonus. MPX — also from Ram

GCRS 5-10 EVENT RESULTS

		FAIRYWEIGHT	ANTWEIGHT	BEETLEWEIGHT
GCRS 5 –	1st:	N/A	Verbal Irony	Voodoo Magic
	2nd:	N/A	Devastator	MPX
	3rd:	N/A	Hoodoo	Walla Walla
GCRS 6 –	1st:	N/A	Platypus	Cake
	2nd:	N/A	Executive Privilege	Voodoo Magic
	3rd:	N/A	Ting Tang	Ramvac
GCRS 7 –	1st:	Nano Nightmare	Shazbot	Leftie
	2nd:	Big Dog's Brother	Cannibal	Voodoo Magic
	3rd:	BluTank	Top Gun	Ramvac
GCRS 8 –	1st:	Berserker	Shazbot	Leftie
	2nd:	Pissed Off Unicorn	WhipperSnapper	Ramvac
	3rd:	Moustache	Platypus	Wallop
GCRS 9 –	1st:	Invertigo	Shazbot	Ramvac
	2nd:	Dirty Sanchez	Thorncupine	Voodoo Magic
	3rd:	Nano Nightmare	Capricant	(round robin)
GCRS 10 –	1st:	Invertigo	Capricant	Ferd
	2nd:	Berserker	WhipperSnapper	DeJaVoodoo
	3rd:	Dirty Sanchez	Verbal Irony	Ramvac

Robotics — uses an activated pincher to grab its opponent and drag them toward the drop-out zone.

GCRS provides a great crowd for South Florida builders to get together and compete frequently. The Ram Robotics team makes trophies for the winners, The Robot Marketplace gives gift certificates, and even Fingertech Robotics gave Silver Spark motors to the winners at GCRS 8. GCRS is always great fun, and the perfect excuse to keep redesigning and improving our bots. **SV**

Shazbot, Top Gun, Thorn, Black & Decker RobotWrecker, Platypus, and a brave little R/C car get ready for the Antweight rumble.

EVENTS

Upcoming Events for July

Scheile Museum Clash of the Bots 3 will be presented by the Carolina Combat Robots. It will take place in Gastonia, NC on July 14, 2012. For further information, go to **www.carolinacombat.com**. **SV**

INTERVIEW

Tool City Robobots Leverages Combat for Learning

● by Chris Olin

For the past six years, middle and high school students from across Crawford County, PA have been building ever-increasing powerful 15 pound combat robots to compete in the annual Tool City Robobots competition hosted by the Crawford County School District and National Tooling and Machining Association (NTMA) of North West Pennsylvania. This past April, 32 teams from 19 schools competed in the most action-packed hard-hitting tournament yet. After the smoke and carnage had been cleared, *SERVO* sat down and spoke with the facility advisors from two of the leading schools in this program.

Chris Yost has been the head of the Cochranton High School Robobots program since 2008. He has a bachelor's degree in Science and Technology, and a teaching certification in Technology Education/Applied Engineering and Technology from the California University of Pennsylvania. He also has a Master's degree in Education Leadership from Edinboro University. He is SolidWorks CSWA certified, and has extensive knowledge of precision machining, parametric modeling, Finite Element Analysis (FEA), construction technologies, digital electronics, graphic and multimedia design, and mechatronics.

Nick Krasa has been teaching Applied Engineering and Technology at Meadville Senior High School for seven years. Over the past six years, the Meadville program has built

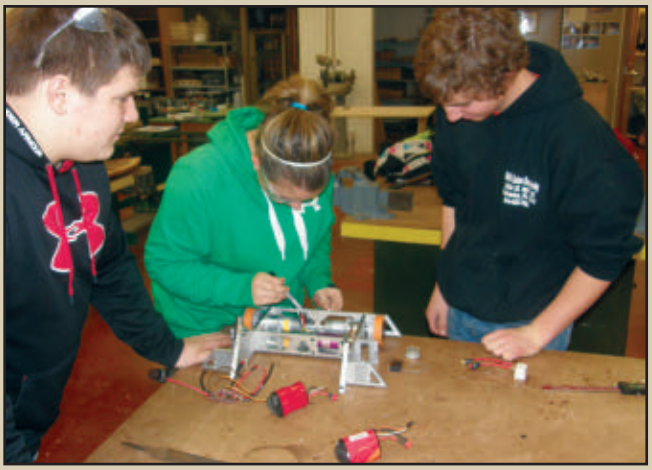


PHOTO 1. The Cochranton High School students troubleshooting a robot drive system.

PHOTO 2. Chris Yost with student design team developing their robot in SolidWorks.

19 robots and won 10 of the 48 trophies given out. This year, Nick has 35 students involved in the building of four robots. Seventeen of his students are seniors who are planning on attending college for engineering. Four of those seniors are in the running for valedictorian of their class.

SERVO: *First, can you give us a quick walk-through of how your programs work, and what concepts you are trying to teach your students?*

YOST: My robotics program at Cochranton High School starts with students learning basic and advanced technical drawing/parametric modeling skills in a Fundamentals of Drafting/SolidWorks class. In this setting, we overview everything from auxiliary view creation to tolerance. From there, students can enter the Robotics Engineering/SolidWorks class in 10th grade. Here, I incorporate such things as parallel and series circuitry in DC electronics, differences between types of motors, advanced parametric modeling, Finite Element Analysis, brainstorming, organization, and mechanical engineering principles such as torsion, shearing, and load distribution, and machine operation using vertical end mills and CNC.

We also discuss and incorporate such science principles as rotational inertia, velocity, and moment of inertia in weapon design. My goals are to create a strong background in design, engineering, science, math, technology, and manufacturing, and provide that "spark" in the student's lives to interest them in possible career fields.

KRASA: I run my program as a student club known as the Student NTMA, where we have a panel of officers who oversee all the operations including the different events we go to, the fundraising, and the distribution of funds to each team. Each team (min of six, max of 10) has a captain, a draftsman, and a production engineer. They all take part in the basic design. Once it is fully designed, they then delegate the fabrication to different individuals on their team who are capable of making the parts. For some of the parts, if we are not able to fabricate them here, I have each team paired up with a local tool shop mentor that will allow the groups to come to their shop and enlighten them as to how things are done in the industry.

SERVO: *Mr. Yost, I see that*

PHOTO 3. Cochranton High School students with their robot, Shockwave.

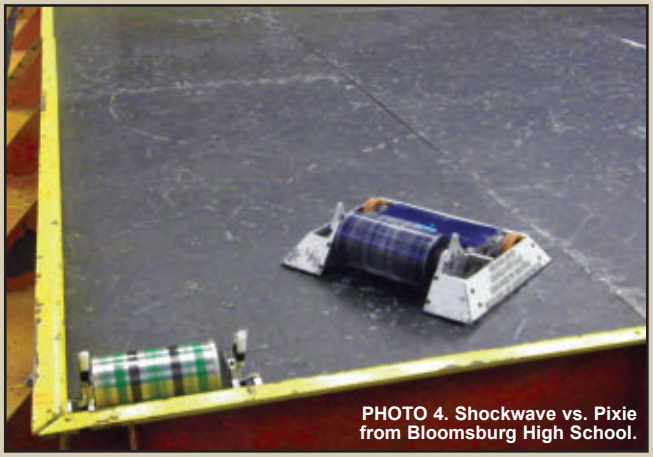


PHOTO 4. Shockwave vs. Pixie from Bloomsburg High School.

your school's three robots took first, second, and fourth place in the documentation contest. What makes for good documentation?

YOST: A great documentation in my eyes revolves around detail. My students document every step of their brainstorming, thumbnail sketches, technical drawings, daily reflections and summaries, electrical diagrams, mathematics, measurements, and finite element analysis (simulation). Students must generate material lists with quantities and pricing for all framework, weaponry, fasteners, electrical components, electrical connectors, and drive components. I tell them that from their portfolio, I should be able to visually and physically be able to reproduce their combat robot from scratch, which means individual drawings that meet ANSI standards must be created along with machining processes. Finally, time is the most important factor. The students put in grueling hours documenting reflections, summaries, drawings, simulations, renderings, and various other components of their portfolios.

SERVO: *Mr. Yost; I also see your robot finished Genocide third overall and Shockwave came in fifth. Any secrets to that success?*

YOST: I feel that their success comes from first off making learning fun for the kids, but also integrating science and math concepts into the construction processes. One important factor I teach the students in weapon design is rotational inertia which is a basic science concept. Without understanding that principle, the students would not be able to design an effective weapon. I also feel that computer-aided simulation and design are a huge part of their success. Before you create, you must be able to design and know that your design is sound. The kids run numerous simulations on individual parts and assemblies, testing different materials and loads on certain areas where the stress on the objects will be at its highest.

From there, the students make important design changes that can withstand a weapon hitting a 15 pound chunk of titanium or aluminum spinning at 11,000 RPM and generating forces in excess of 16,000 ft lbs. Overall, the most important factor is the time and dedication. I feel that the time and dedication my students have to their work is the main reason they succeed. They believe in themselves and that is what makes a confident individual.

SERVO: *Mr. Krasa, I see that your robot Ripsaw took "Best Engineered" and Dr. Destruction PhD took "King of the Ring." How are those awarded?*

KRASA: This is the fourth year in a row that we have won "Best Engineered." Every year, there are four judges (different industry leaders in the area) that walk around and evaluate the bots throughout the day, and vote on which one they thought had the most thought put into it. We actually took first and third in that category this year.

King of the Ring is voted on by the judges and the event MCs. It usually goes to the most destructive bot entered into the competition and although Dr. Destruction had done well during the competition, they did exceptionally well after they were eliminated in the Rumbles where there were five or more bots in.

SERVO: *Some of the concepts you talk about are normally second or third year college work. Doesn't that seem a bit much for high school students?*

YOST: Yes and no. I think that some of the concepts being taught are right on par for students at this educational level, being that they have been introduced to the concepts and I am simply diving further and having them use these concepts in an application-based setting. As for the concepts that are above their educational perspective, I would rather have them be introduced to them now and ask questions about them than see them later in a college setting where they may not have as much one-on-one time with a

PHOTO 5. Obligatory kids with messy pit table picture.

PHOTO 6. Dr. Destruction PhD in the pits.

professor to understand them.

KRASA: At this point, we have created 19 different robots from scratch, all of which were designed by high school students. With the right motivation, high school kids will amaze you with their abilities.

SERVO: *Are there any academic requirements for a student to be a part of your program? Do you have a waiting list of students wanting to participate?*

YOST: Absolutely. In order to qualify for the Robotics Engineering/SolidWorks class, students must pass Fundamentals of Drafting/SolidWorks with an 80% or higher through the year, plus receive a recommendation from me in order to qualify for the class. My class caps off at 12 students due to the nature of the course, but I have had so much interest from parents and students who have taken the class this year that I am in the process of writing curriculum for a Re-Engineering and Design/Robotics course.

SERVO: *All of this seems very cost-intensive and benefits a fairly small number of students. Is it really worth it?*

KRASA: Absolutely! As long as the program gets a couple of kids a year interested in entering the technical workforce, then it was well worth it. The whole point of the program is exposure and with the skilled labor shortage in the US, I think it's a great investment. Aside from a few of the electronics, the majority of the parts can be reused from year to year.

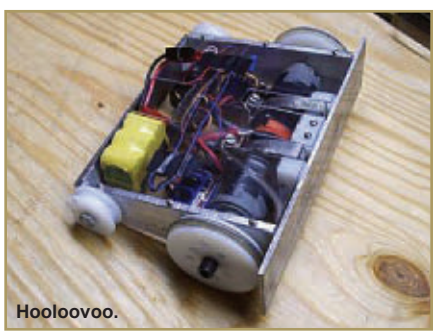
SERVO: *What advice would you give to someone looking to start a program like this?*

YOST: I would suggest finding someone running a program similar to what you are planning on doing, and spending time with them going over the concepts that need to be taught and how they manage their time with it. A background in parametric modeling whether it is in SolidWorks, Pro/Engineer, or AutoDesk Inventor is a must. Next, start as early as possible. Time is extremely important due to the fact that you're trying to teach students new to this the entire engineering design phase from start to finish. Finally, let the students do some destructive testing. Their designs are going to be hit, smashed, and thrown through the air. Find out the problems before their competition begins and give them plenty of time to fix those issues. SV

The History of Robot Combat: Rise of the Insects Part 2

● by Morgan Berry

In the April issue of *SERVO*, we told the story of the development of the Antweight class in the United States. This month, we'll discuss their larger counterparts: the Beetleweights. These bots weigh in at three pounds and have long been popular with new and veteran builders alike. The popularity of the bots lies in the simple fact that you can put a lot more stuff into a three pound bot than you can on a one pound. Since Beetleweight motors are not much larger than Antweight motors, the builder is left with some available weight to add weapons and other extras. A three pound bot was a natural step between the tiny one pound Ants and the larger 12 pound Hobbyweights, and the weight class spread quickly throughout North America.



Hooloovoo.

Beetleweight fights seem to have originated in the northeast United States but through the vast online network of builders, quickly spread throughout the country. The first documented Beetleweight match was in October 2002, hosted by the Lazy Toad Robot Club in Pennsylvania. The two Beetleweights in attendance out of 68 total bots were Hooloovoo, from Team Tier Robotics, and Yattering, from Team Dragon Combat Robots. Hooloovoo was a plow, while Yattering was a spinning drum. By April 2003, it had spread to Seattle, WA and High Springs, FL. In May, Beetleweights were in Texas, and by mid 2003, Beetle matches were being held at almost every competition in the United States and Canada.

The early bots were simple. At the High Springs Antweight Challenge, for example, the bots in attendance were modified RC toys. Soon, Beetles became more sophisticated; The Robot

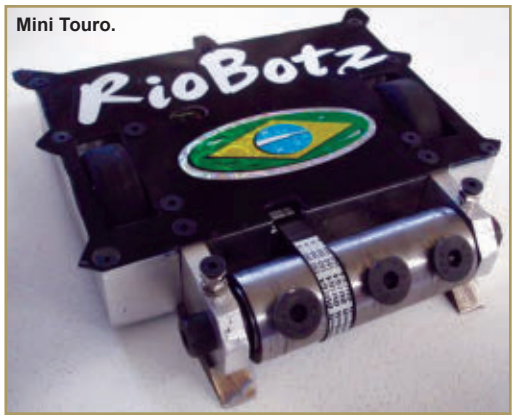
Marketplace and other online retailers began selling motors and parts specific to Beetleweights. Just a year after the Beetleweight class took off, for example, Kevin Berry's Fir Darrig Beetle had its previously impenetrable 1/8 inch aluminum cut through by Team Wyachi's 3A. Kevin quickly updated his armor to titanium to catch up with the growing power and complexity of the Beetleweight bots.

Builders liked the freedom that the added weight allowed over the Antweight class. A few ounces dedicated to the batteries or motor affected the overall design of the bot much less than they would in an Antweight bot. Of course, any remaining unused weight could go into more fun extras.

Beetleweights often feature complex motorized weapons. The added power that the extra weight allows makes for interesting and fast paced fights. They were also larger than Ants, which allowed them to be less compact and thus easier to build than their smaller counterparts.

Today, the Beetleweight class is still active, although not as frequently as in the early 2000s, which is considered the golden age of Insect fights. RoboGames — which is by far the largest current competition in robot combat — had 32 Beetleweights registered for the April 2012 competition.

A growing trend among the Beetleweight class is autonomous bots. These bots are not remote controlled, but instead use sensors to track and fight another autonomous bot. An example of these futuristic bots is Mini Touro+ by team RioBotz. Dr. Marco Meggiolaro — author of the famous *Combot Tutorial* — designs and builds the Touro series along with students from his university in Brazil. The bots utilize a spinning drum on the front of the robot. The autonomous Mini Touro+ — along with the RC versions in every weight class by the same name — is considered one of the top competitors in the weight class.

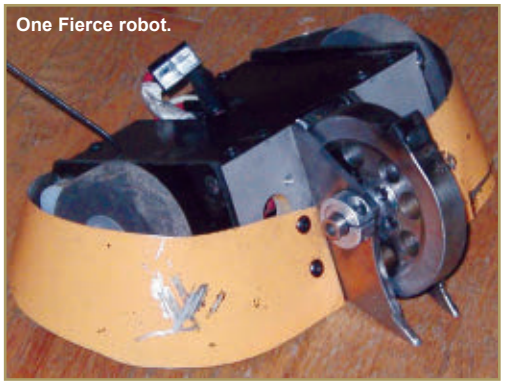


Mini Touro.

Another powerhouse in the Beetleweight class is Gene Burbeck of Fierce Robotics with his "One Fierce" series of Beetleweights. He is often considered one of the best and most prolific Beetleweight builders in the world. His bots are all spinners, heavily armored, and have a unique feature in the building world: they are reliable and always work.

Beetleweights, although a few years younger than their Antweight counterparts, have risen to major prominence in the robot combat world, and may have even become more popular. The appeal of the Beetle as a middle sized bot between the one and 12 pound weight class continues to be widespread. Beetles are a popular weight class today, and probably will be for years to come.

Next month in this series, we'll cover robot combat in the northeast. **SV**


One Fierce robot.

A Decade of Robot Fighting

● by Andrea Suarez

The box is locked, the lights are on ... it's robot fighting time! I had seen the BattleBots TV show on Comedy Central as a kid, but the rush of my first three minute match would have a bigger impact on my life than I could have ever known at that moment.

I have spent the last 10 years building robots weighing from 150 grams to 120 pounds, and now that I'm getting ready to face the "real world," I realize that this hobby has shaped so many facets of my life. It dramatically redirected my career choice to biomedical engineering and helped me secure my first job as a research and development medical device engineer. It has introduced me to many great friends and to my boyfriend and fellow robot builder, Mike Gellatly. I went from having never left South Florida, to travelling around the country to AL, MN, NY, CA, IL, and all over FL.

Robotics has taken me for a walk across the Golden Gate Bridge, on a 20 hour bus adventure to Alabama, a tour of the Jelly Belly Jelly Bean Factory, a white-water raft ride on the Niagara River rapids, on my first airplane, to the top of the Sears Tower, and many other remarkable experiences.

I thought the end of college

2002: My first robot competition. Left to right: Carolina de Freitas, Elizabeth Turner, Traci Walder, Anna Baez, Andrea Suarez, and Briana Baker.

2003: Learning to weld at StarBot.

2004: Our first BattleBots event in MN in 2004.

2005: Lonnie Quinn interviewing our Mean N' Green team for NBC 6. Marvinator's shell is still in the process of being made.

2006: A few of us celebrate our 18th birthday along with our second place win at BattleBots 2006.

marked the end of my robot fighting days, and I tried to convince myself that it was time to outgrow the sport. I remembered high school orientation at Carrollton School when I was 13, as I stood in the chemistry lab looking over the six lab tables that formed a fighting area to contain a hand-painted metal monster. The girls stood behind the controls showing off their creation and I immediately knew I had to get involved.

Over the next few months, our evenings were spent in the biology lab building our first robot — a tabletop task bot. We decorated our new creation, made t-shirts that said "Girls Build Bots Better," and showed up to our first competition — and we WON! We were hooked.

We spent many evenings and weekends at StarBot — a local non-profit robotics shop — as we learned to use a mill, bandsaw, lathe, drill, welder, etc. The satisfaction of drawing something on a piece of paper and being able to walk into the shop and make it was addicting. As 14 year old girls, the excitement of using power tools and heavy machinery overpowered any intimidation from the flying sparks. I came home with tiny holes burnt into my Carrollton School uniform skirt and grease on my shirt, but we were determined to keep up with the boys and prove ourselves with our robot.

After building a few more task bots and 15 lb BattleBots, it was finally time to build our first 120 lb BattleBot as high school sophomores. Kerminator slowly came to life with a pentagon-shaped welded steel frame and a red saw "tongue of death" between two foam light-up eyes. Our team of eight girls — Mean N' Green — flew to Owatonna, MN and was quickly intimidated by the display of robots at BattleBots IQ 2004. We only won two matches, but Nola Garcia (CEO of StarBot) saw some potential in our team and paired us up with a former BattleBots team, Checkmate, to mentor us as we built our next robot.

Marvinator became an octagonal full-body spinner that competed for the next two years. BattleBots 2005 was fairly successful, but our luck ended suddenly with an explosion of our homemade battery packs. BattleBots 2006 marked our senior year of high school and a complete re-design of Marvinator, taking into account everything we had learned (the hard way) through the previous design.

With each win, we became a little more confident than the match before, and we worked frantically between matches with a blowtorch and a hammer to repair the damage. We slowly worked our way up the bracket, destroying beautifully machined robots and veteran teams with our handmade machine (and a little luck, of course).

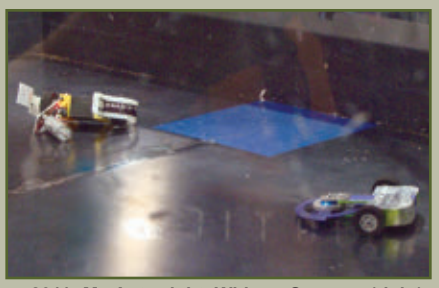
The night before the final matches, it was time for Senior Grad Nite at Disney World with our graduating class. We finished our matches for the day and took the four hour bus ride to spend the night riding roller coasters, watching live bands, and having a blast. We

2007: Mike and I share the Coolest Robot award after our first competition together.

2008: Team Rhino (left) from Puerto Rico has been our rival in the arena year after year, but after the buzzer we have great friends that took us sailing, hiking through El Yunque, and exploring caves on our visit to Puerto Rico.

2009: Category 5 (right) vs. Bradley University's Jailbreak in CA 2009. We ran into each other again at Mecha-Mayhem 2011 in IL for an Insectweight event!

2010: Mike and I competed against each other at BotsIQ 2010 for one of the most intense matches of the event.



2011: My Antweight, WhipperSnapper (right), competes in Mecha-Mayhem 2011 in IL.

returned just in time for our first match the next morning, and what followed was a day so intense that we didn't have time for exhaustion. We finished the event in second place — a record for our school and the storybook ending to our high school robotics career.

Robotics in college was an entirely different experience in many ways. I went from an all-female team to being "the girl" on the team, but the guys were glad to include me and even let me drive the robot. A lot of manual machine time was replaced with CNC and waterjet cut parts, and I learned how to apply the engineering principles I was learning in class to avoid a lot of trial and error.

I eventually became president of our college robotics team and travelled to Vallejo, CA in 2009 where BattleBots filmed a mini-series that included our third place win. Some small changes earned us a second place win the next year — and four Coolest Robot awards along the way. I stayed in school an extra year to get my Master's degree in Biomedical Engineering (or was it to compete in BattleBots IQ one last time?), and 2011 brought another second place victory, along with the award I had coveted most throughout the years: the Best Engineered Robot award.

It was an incredible experience and I wasn't ready to outgrow the sport quite so soon, but we no longer had access to a school's budget. Mike and I decided to get together with a few South Florida builders and try our hand at much smaller Insectweight bots. Going from designing 120 lb robots to designing 1 lb robots had quite a learning curve as we discovered what parts and materials to use. It was like starting from the beginning all over again. These small bots have turned out to be really exciting and are surrounded by a great group of competitors of all ages and backgrounds. I can't wait to see where this crazy sport will take me next. In addition to competing any chance we get, we stay involved by hosting exhibitions at local schools, judging events, and we are in the process of planning our own event for the first time.

This hobby has become a way of life and I am constantly surprised at the camaraderie and enthusiasm that surrounds everyone involved in the sport. Many thanks to those that have made this possible for me throughout the years. Let's make sure kids can keep building bots! SV

A Radio For Robots

by Fred Eady

Discuss this article in the *SERVO Magazine* forums at http://forum.servomagazine.com.
www.servomagazine.com/index.php?/magazine/article/june2012_Eady

Robots and radios. A pair made in robot heaven. I'm always on the beat looking for new and unique stuff to use in the real world. When I find something that's truly useful and it really works, it ends up filtering out through my fingers onto the printed page.

The editorial goo that formed this month's discussion is provided by RF Digital. Our discussion will revolve around a radio module that can act as an RFID device, a key fob, and a wireless serial link. All we have to do is choose the mode of operation and flip the power button.

The RF Digital RFD21733

The tiny radio module I've captured in **Photo 1** is an out-of-the-box ready to use fully FCC compliant 2.4 GHz transceiver. This particular radio is designed to be highly tolerant to noise and other 2.4 GHz interference. In that the RFD21733 can operate in noisy motor environments and fend off 2.4 GHz Wi-Fi traffic, it is perfectly suited to robotic applications. The robust operation of the RFD21733 is attributed to the built-in RFDP8 interference-tolerant user application protocol. RFDP8 is inherent to the RFD21733 and transparent to the user.

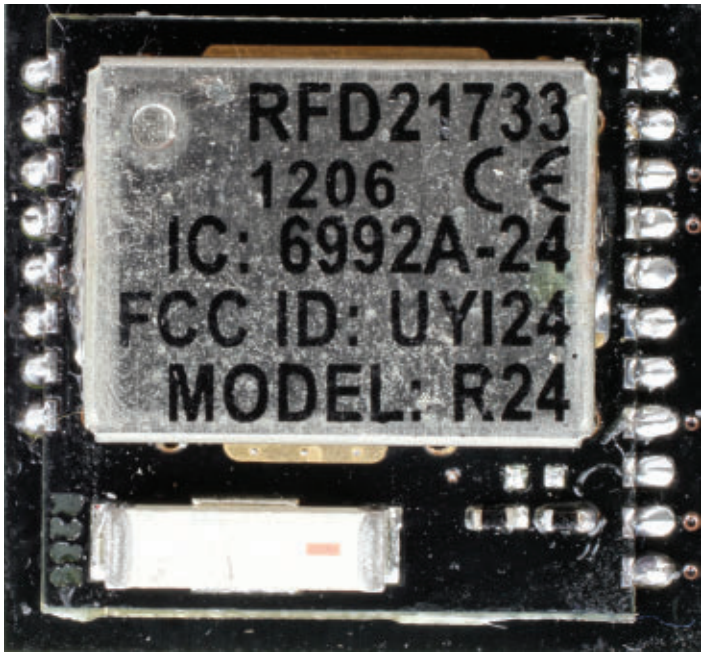


PHOTO 1. The RFD21733 is a complete radio transceiver subsystem and includes an on-module antenna.

The RFD21733 can be coaxed into RFID mode (Mode 0), Logic Switch Transmitter/Receiver mode (Modes 1, 4, 5, 6, 7), or act as a 9600-8N1 Serial UART Transceiver (Mode 2, 3). If that's not enough, the RFD21733 is network capable, as well. Modes 3, 5, and 7 are the network mode variants. To support the various network modes, a unique 32-bit ESN (Electronic Serial Number) is assigned to every RFD21733.

The kicker is that we don't need a microcontroller or complex firmware to get to all of that functionality. It's all available with the push of a button. Our RFD21733 radio module is actually mounted on an RF Digital RFD21737 evaluation board. All of the pushbuttons, DIP switches, LEDs, and logic we need to put the RFD unit through its paces are packed on the evaluation board. The entire evaluation board complete with an RFD21733 transceiver is shown in **Photo 2**.

The evaluation board includes a three-position DIP switch that is used to select the RFD21733's mode of operation. There are three pushbuttons that allow their logic levels to be transmitted in three-Input Switch Logic Transmitter mode (Mode 1). Each pushbutton is complemented by an LED that displays the transmitted

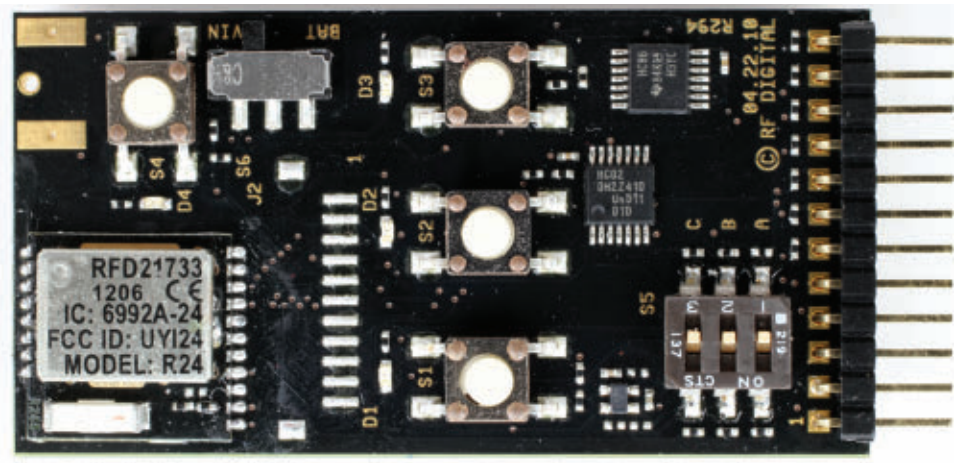


PHOTO 2. There are four good reasons for this evaluation platform. It works out of the box, we don't have to do any programming, and we don't have to design any hardware or fabricate any circuit boards.

logic in the Three-Output Switch Logic Receiver mode (Modes 4, 5, 6, 7). The three RFD21733 I/O pins that are under the three pushbuttons are brought out to the evaluation board's 12-pin header. The same holds true for the board's LEARN pushbutton and associated TX/STATUS LED. To accommodate clean mode changes, the RFD21733's RESET pin is also brought out to the header. The mode selection inputs are only read immediately after power-up. Using the reset pin forces the RFD21733 to reset and perform the mode input read operation without dropping power to the radio.

An onboard 3.3 volt voltage regulator allows the evaluation board to be powered from 5.0 volt power sources. There's a CR2032 battery on the flip side of it that provides long-term power for RFID operation.

RFD21733 Modes

If you need to send data from point A to point B, put a pair of RFD21733 radios into Serial Transceiver mode (Mode 2). Serial Transceiver mode forms a bidirectional half-duplex link. Switching from transmit to receive mode is automatic; the rule being that transmission takes priority. So, as long as you're transmitting you don't have the ability to receive.

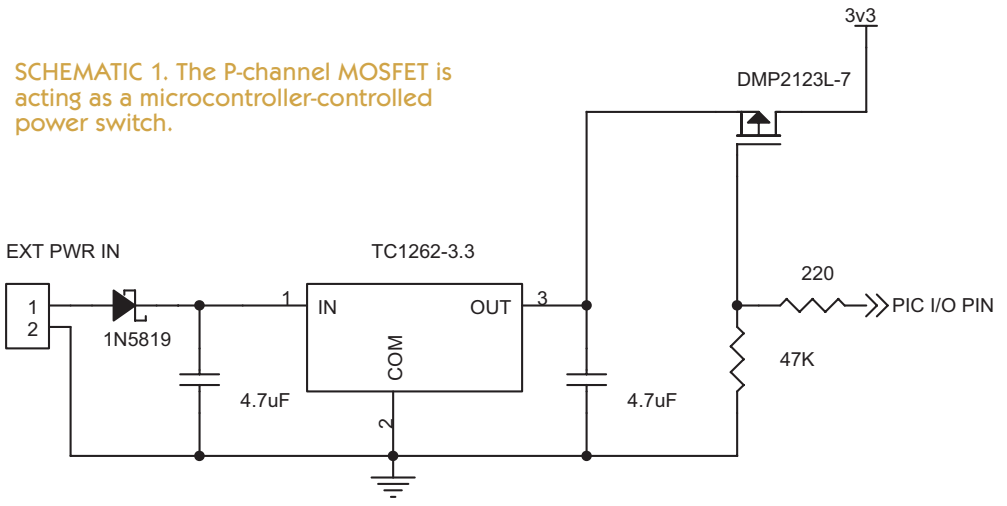
When the transmitted data is broadcast, any RFD21733 in range will receive the transmission. You can be selective with your messaging by entering Network mode. In Network mode, only data from transmitters with known ESNs is accepted.

Network mode for Serial Transceiver mode is Mode 3. Mode 5 is the networkable version of Three-Output Switch Logic Receiver mode with a 500 mS hang time, while Mode 7 is Three-Output Switch Logic Receiver mode with a 20 mS hang time. Hang time, in this case, is the output pulse width. The complete RFD21733 Mode chart is shown in **Figure 1**.

To enter Network mode, an RFD21733 must learn the ESN of any module from which it intends to accept transmissions. The LEARN signal is an active high input. To enter LEARN mode, the LEARN pin of the RFD21733 must be pulsed logically high for a minimum of 20 mS before being returned to a logically low level.

RFDP8 Standard Mode Chart © RF Digital Corp. 07.05.11 10:07 PM		Mode Select Inputs							
		2	1	0				Learn/Status	
Mode	Description								
0	Active RFID Transmitter	0	0	0	IN 3	IN 2	IN 1	TX LED	
1	3-Input Switch Logic Transmitter	0	0	1	IN 3	IN 2	IN 1	TX LED	
2	Serial UART Transceiver, 9600, N, 8, 1	0	1	0	TXD IN	RXD OUT	LOGIC I/O	X	
3	Serial UART Transceiver, 9600, N, 8, 1	0	1	1	TXD IN	RXD OUT	LOGIC I/O	ESN LEARN	Network
4	3-Output Switch Logic Receiver - 500 ms	1	0	0	OUT 3	OUT 2	OUT 1	X	
5	3-Output Switch Logic Receiver - 500 ms	1	0	1	OUT 3	OUT 2	OUT 1	ESN LEARN	Network
6	3-Output Switch Logic Receiver - 20 ms	1	1	0	OUT 3	OUT 2	OUT 1	X	
7	3-Output Switch Logic Receiver - 20 ms	1	1	1	OUT 3	OUT 2	OUT 1	ESN LEARN	Network
Module RFD21733 / RFD21735 Pin Number:		3	17	16	7	6	5	4	
RFDANT RFD21742 / RFD21743 Pin Number:		7	6	5	11	10	9	8	

FIGURE 1. The RFD21733 modes make more sense when laid out like this.



SCHEMATIC 1. The P-channel MOSFET is acting as a microcontroller-controlled power switch.

Upon entering LEARN mode, the LEARN pin reverts to output mode and is driven logically high. With a bit of buffering, the LEARN pin can be used to drive an LED in this state. The RFD21733 module will learn — or store — the ESN of the first module that transmits a packet that reaches its receiver. The data within the LEARN transmission is discarded while the incoming ESN is stored. Every packet transmitted by an RFD21733 module contains the transmitting module's ESN. Once the ESN is stored, the receiving unit toggles the LEARN pin three times. If an ESN is not received within 10 seconds of entering LEARN mode, the receiver will drive the LEARN pin logically low and revert the LEARN pin to input mode.

A maximum of 60 ESNs can be learned by a receiver. To clear the ESN list, the LEARN pin is driven logically high for a minimum of 10 seconds. To indicate the ESN erasure is complete, the RFD21733 will toggle the LEARN pin very quickly for a few seconds.

Network mode can be applied to a single pair of RFD21733 nodes (peer-to-peer), one to many nodes (multi-point), and many to many nodes.

The most basic of modes is the RFID mode (Mode 0) in which a transmitter sends a packet containing its 32-bit ESN every two seconds. The receiving node — which is operating in Mode 4 — toggles its OUT1 pin low to high for 500 milliseconds (20 mS if operating in Mode 6) with the reception of every RFID packet.

An interesting variant of the RFID mode has the receiving unit convert the incoming RFID packet into five bytes that exit its RXD pin as a 9600-N81 serial stream. Byte 1 of the RFID packet contains 0b00010001 with the remaining four bytes carrying the transmitter's ESN. This is accomplished by forcing the receiver into Serial Transceiver mode (Mode 2).

I associate the Three-Input/Output Logic Switch Transmitter/Receiver modes with key fobs. The transmitting RFD21733 — operating in Three-Input Logic Switch Transmitter mode — remains in deep sleep until a logical high is applied to any combination of its inputs (IN1, IN2, and IN3). The receiver — operating in Three-Output Logic Switch Receiver mode — echoes the state of the transmitter's inputs on its output pins (OUT1, OUT2, and OUT3).

Like the RFID/Serial Transceiver mode combination, the Three-Input Logic Switch Transmitter mode (Mode 1) can be coupled with a receiver in Serial Transceiver mode (Mode 2) to convert the incoming packet into a 9600-N81 serial stream. The most significant four bytes still contain the transmitter ESN. However, the least significant byte contains the input logic states of the transmitter's IN1, IN2, and IN3 inputs in bits 4, 5, and 6, respectively.

This whole mixed mode thing can be inverted. An RFD21733

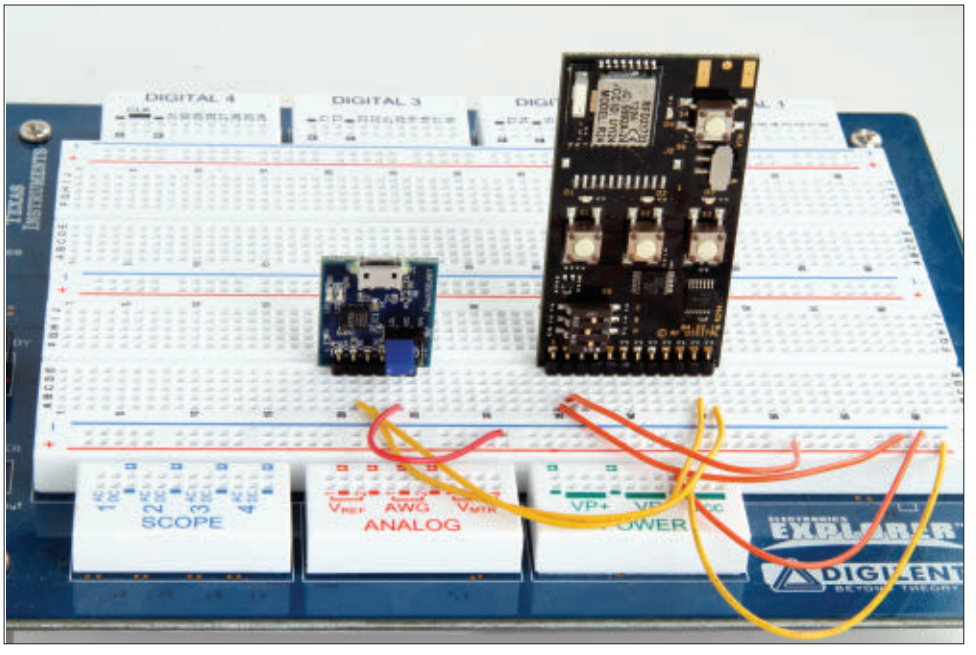
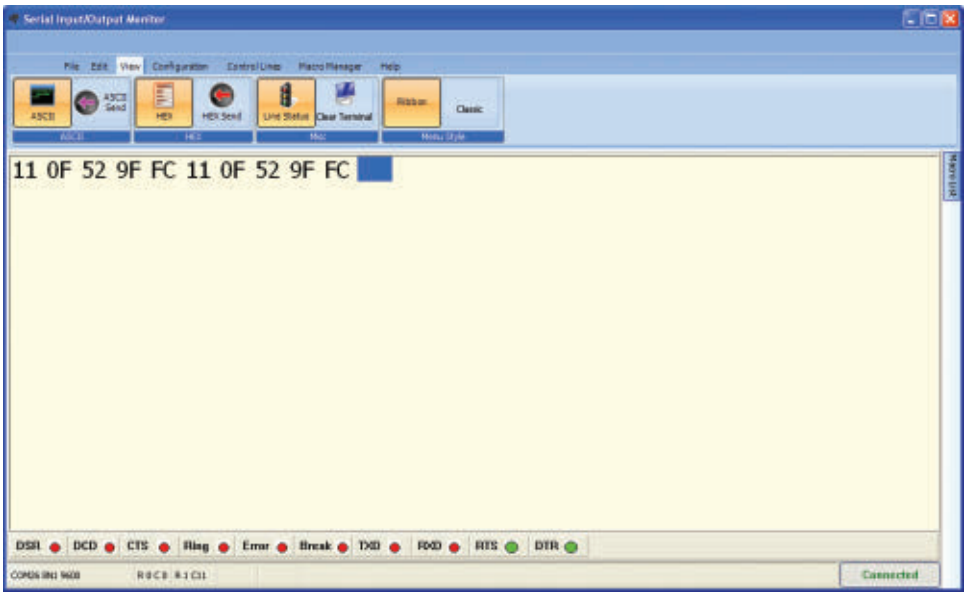


PHOTO 3. The Digilent Electronics Explorer is an all-in-one development tool. In addition to the power and breadboard, the Electronics Explorer is supported by Waveforms software that puts volt meters, an oscilloscope, a waveform generator, and a logic analyzer on the breadboard.



SCREENSHOT 1. SIOW is a terminal debug application that is part of the CCS C compiler package. There are multiple packets displayed. The first byte of each packet is what we expected (0x11). Are the rest of the bytes really the transmitter's ESN?

node in Serial Transceiver mode can send its INx logic levels to a receiver in Three-Output Logic Switch Receiver mode. Instead of processing the incoming data through its RXD serial output, the receiver will echo the transmitter's INx logic levels on its OUTx pins. In addition, Network mode can be applied to every mode combination we've touched on.

RFD21733 Power Considerations

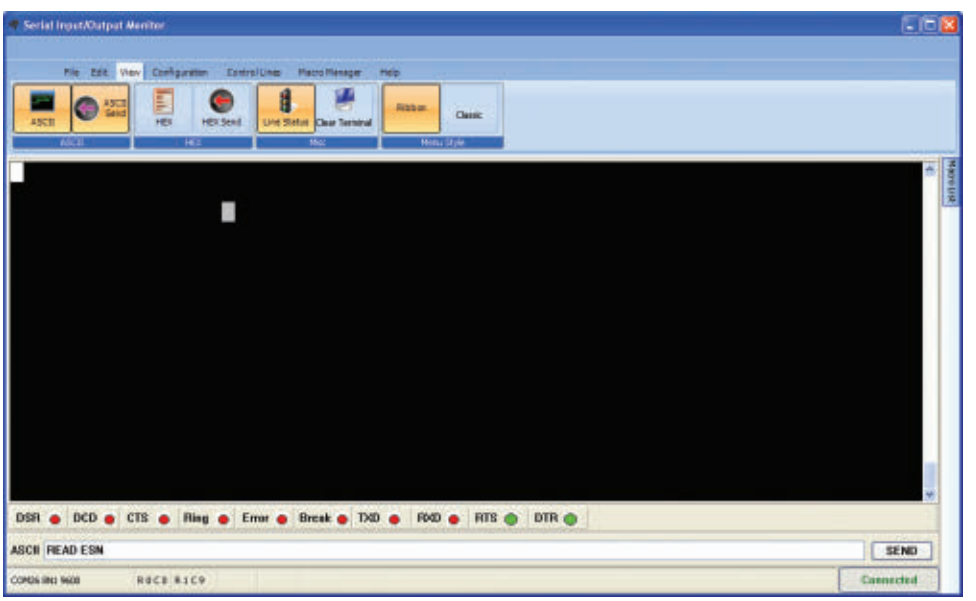
With the receiver powered up, the RFD21733 draws 17 mA. Transmitting RFD21733s pull 14 mA. The unit can be ready to operate as a transmitter or receiver in as little as 3 mS following power-up. Thus, a transmitting or receiving RFD21733 radio can be turned totally off until it is needed. All you need is a P-Channel MOSFET circuit like the one I've devised in **Schematic 1**. As long as the MOSFET gate is driven logically low, power will flow to the target. Forcing the MOSFET gate logically high with a PIC I/O pin turns the MOSFET off and deprives the target of power.

Firing Up Our RFD21733

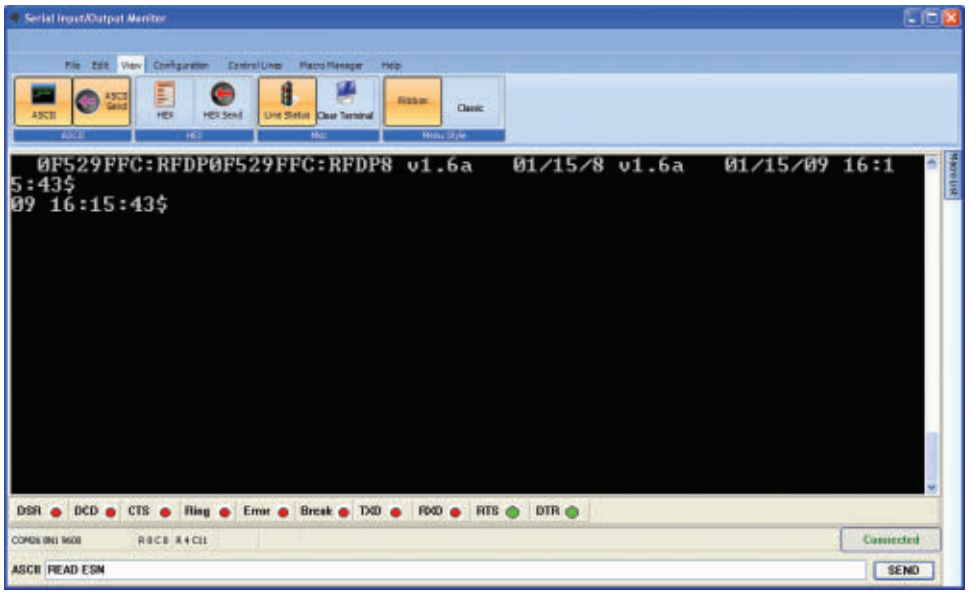
In all honesty, I've already flipped the power switches on my pair of RFD21733s. With a simple DIP switch change and a power cycle, I was instantly able to see the RFID mode in action. I then switched to Three-Input Logic Switch Transmitter mode on the transmitter and Three-Output Logic Switch Receiver mode on the other RFD21733. Pressing the pushbuttons on the transmitter illuminated the associated LEDs on the receiver. That kind of stuff is hard to photograph with a still camera. So, let's use a tricky little terminal program to take a look at an RFID packet.

My RFD21733 RFID packet capture setup is shown in **Photo 3**. I used a Digilent Electronics Explorer to supply 3.3 volts to the RFD21733 and provide a connection platform for a Digilent Pmod. The Digilent PmodUSBUART is wired to the RFD21733's TXD and RXD pins. However, the PmodUSBUART gets its power from the USB connection. The RFD21733's TXD pin is an input, while its RXD pin is an output. With that, the PmodUSBUART's TXD pin — which is an output — connects to the RFD21733's TXD pin. Obviously, both the RFD21733 and Pmod's RXD pins are electrically connected.

We're expecting a five-byte packet that looks like this: 0x11 0xAA 0xBB 0xCC 0xDD. Bytes 2 through 5 are the transmitter's ESN. To see the hexadecimal data, we'll need a terminal emulator that can serve up a hex display. That counts HyperTerminal and TeraTerm Pro out. A really neat terminal program that can display hex data is part of the



SCREENSHOT 2. The ability to send and receive both hexadecimal and ASCII characters is only one of SIOW's strong suits.



SCREENSHOT 3. This ESN data was obtained by manually invoking the ESN read-back function.

Sources

RF Digital
RFD21733
RFD21737 Evaluation Board
www.rfdigital.com

CCS
CCS C Compiler
SIOW
www.ccsinfo.com

Digilent
PmodUSBUART
Electronics Explorer
www.digilentinc.com

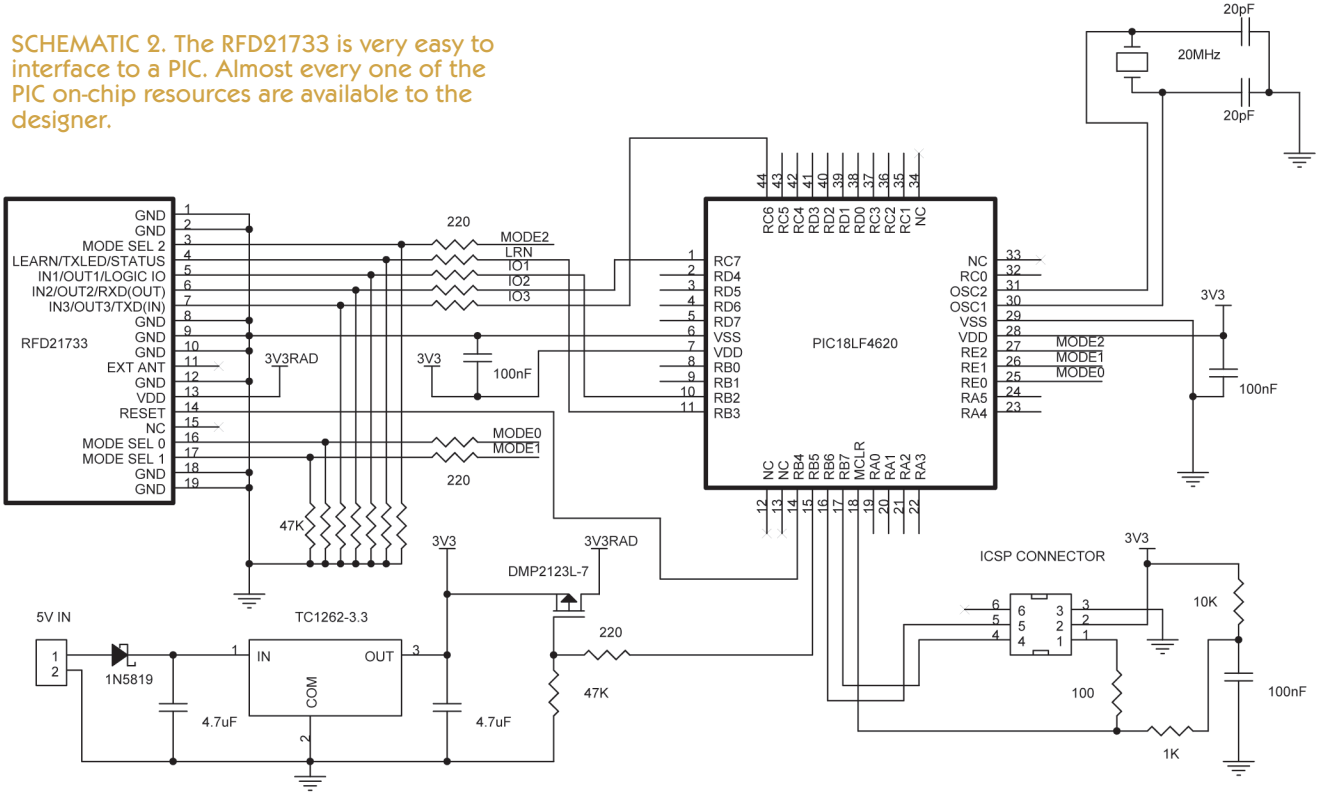
Lemos International Co., Inc.
RF Digital RFD21737
Development Board
RF Digital RFD21733
Radio Module
www.lemosint.com

CCS C compiler. It's called SIOW. After placing the receiver in Serial Transceiver mode and the transmitter in RFID mode, I fired up both RFD21733s.

Well, looky here. We have captured some data in **Screenshot 1**. The first byte is exactly what we expected and identifies the packet as an RFID packet. The remaining four bytes should be the transmitter's ESN.

There is a way we can verify the received ESN. The RFD21733 can be forced to dump its ESN in ASCII format using the ESN read-back procedure. All we have to do is enter a UART mode (preferably Mode 2), tie the RFD21733's LEARN pin logically high, and hold the RESET pin logically low. Since we're doing this without a microcontroller, we will call upon SIOW to send the ASCII text "READ ESN" to the RFD21733 as soon as we release its RESET line. As you can see in **Screenshot 2**, I've primed the SIOW ESN read-back terminal session. As soon as I manually disconnect the RESET pin from ground, I'll

SCHEMATIC 2. The RFD21733 is very easy to interface to a PIC. Almost every one of the PIC on-chip resources are available to the designer.



send the text that I loaded in **Screenshot 2**. Technically, I have less than one second after pulling the ground connection to get the ASCII text to the RFD21733.

I pulled off the manual ESN read-back procedure successfully. **Screenshot 3** verifies that the transmitted ESN packet bytes in **Screenshot 1** match up with the ESN data we obtained with the ESN read-back procedure.

Adding a Microcontroller

The evaluation board contains combinatorial logic that is used to buffer the LEDs and provide some mode switch logic. You can download a complete schematic diagram from the RF Digital website (see **Sources**).

Once you get your copy of the schematic, you'll notice that most of the RFD21733's I/O and control pins are pulled down with 47KΩ resistors. The only exceptions are the RFD21733's RESET pin — which is pulled logically high internally — and the mode input MO — which is internally pulled logically low.

Note also that all of the I/O and control pins are electrically connected in series with a 220Ω resistor. The 47KΩ and 220Ω resistors should also be included in your microcontroller-supported RFD21733 design. As it stands, you can simply connect a microcontroller's I/O directly to the evaluation board's 12-pin header. Only eight I/O lines are required, so the remaining I/O lines of the PIC can be used to interface to other stuff like motor drivers and sensors.

A typical microcontroller-enhanced RFD21733 design is drawn up in **Schematic 2**. The PIC's I/O connections should not be foreign to you. If you program using the CCS C compiler, simple bit I/O functions are all that will be needed to command the RFD21733. Note that you also have program control over the RFD21733's power. Smooth transitions between modes can be accomplished via the RFD21733's RESET line or by a power-on reset. All of the PIC's peripherals are available to the programmer with the exception of the UART which is dedicated to the RFD21733. The external interrupts, PWM, capture-compare, I^2C, analog-to-digital, and SPI subsystems are all free to use along with the RFD21733.

A Robot's Radio

What more could a robot want? The tiny RFD21733 can stand alone powered only by a coin cell. It's just as much at home when dangling from the UART of a PIC. With built-in 16-bit CRC, motor noise tolerance, and networking capability, the RFD21733 is truly a robot's radio. SV

Sounding Off Part 4

by Gordon McComb

Creating a Custom Sound Co-processor

www.servomagazine.com/index.php?/magazine/article/june2012_McComb

Discuss this article in the *SERVO Magazine* forums at **http://forum.servomagazine.com**.

My first fascination with electronics was sound-making gizmos. It didn't matter what sound: sirens, warblers, choo-choo train effects ... you name it, I wanted to build it. Most used a 555 timer IC, but others involved the latest specialty sound-making chips of the day, like the venerable Texas Instruments SN76477 — made famous by the Space Invaders arcade game.

That was then, this is now. For the past several months, we've looked at ways to use the Arduino to create sound — all the way from simple tones, to speech, MIDI, and even MP3 playback. As capable as the Arduino is, sound generation tasks can use up a lot of processing time. The more cycles your robot's brain spends on making noise, the less time it has for driving around and avoiding obstacles.

Sound is an ideal process to offload to another microcontroller. You could always use a second Arduino, but an even better method is to use a microcontroller that is designed from the ground up to handle multiple tasks at the same time.

In this fourth and final installment of Sounding Off, you'll discover how to use the Parallax Propeller microcontroller to augment the audio generation features of your Arduino-based robot — generate tones, synthesize MIDI instruments, even play digitized WAV files. While the emphasis is on sound, know the same general concepts work for any co-processing task, such as operating motors and reading sensors.

Getting to Know the Propeller

Most microcontrollers are based on a design that's several decades old. That's not necessarily bad, but it's not the only way to build one. In the typical blueprint, a central core runs a main program, while separate hardware peripherals handle specialized tasks such as serial communications, counting, timing, and I/O pin interrupts. Whenever a hardware component needs to interact with the central core, the main program is temporarily halted. Once the side process is completed, the main program continues where it left off.

The Propeller is different. First off, it contains not one core, but eight. These cores — called cogs — are each able to simultaneously run its own program.

Rather than rely on a specific arrangement of peripheral hardware, each cog can synthesize any kind of peripheral. This makes the Propeller supremely flexible. For example, if you just happen to need four high speed serial ports, you can define multiple cogs to handle all the communication. On the other hand, if you don't need that many serial ports, there's no wasted hardware. The cogs are free for use as something else.

One of the most touted features of the Propeller is that all eight cogs contain their own internal video generator. We won't be using video in this article, but it's a handy expansion feature for any robot. The Propeller can directly connect to a VGA or composite video monitor. You can

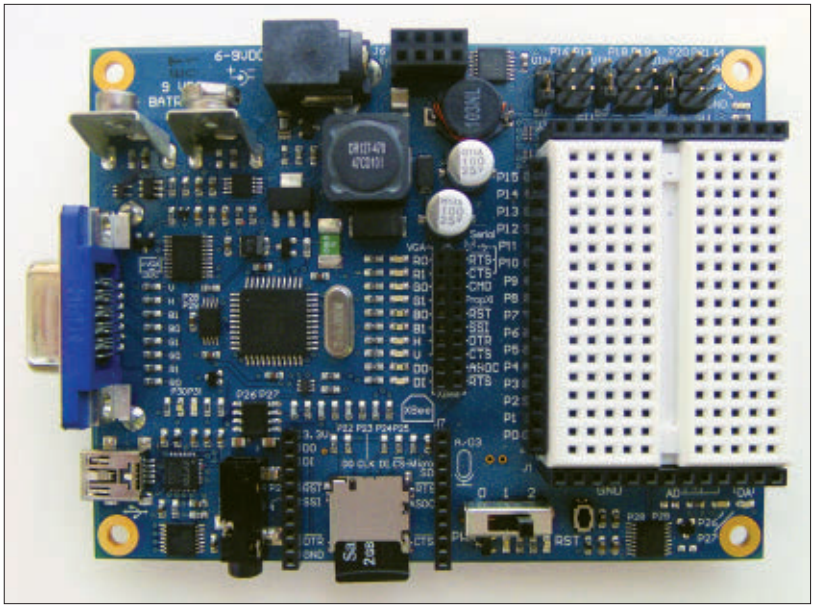


FIGURE 1. The Propeller Board of Education (PropBOE) serves — among other things — as an able experimenter and development board for working with sound.

use the chip's video output to make a big and bright LCD screen capable of color graphics and text.

Getting to Know the Propeller BOE

Parallax offers a number of Propeller-based development boards. Their latest — and one of the most versatile — is the Propeller Board of Education, or PropBOE (see **Figure 1**). This board is not inexpensive, but it combines nearly all of the features you'll need as you develop and experiment with Propeller programs. Its size and form factor match those of the venerable BASIC Stamp Board of Education; both measure 3 x 4 inches, and contain a 10 row by 17 column mini solderless breadboard.

Among the built-in hardware of the PropBOE:

- Both 5V and 3.3V voltage regulators, for supporting three power sources: USB, nine volt battery, or 6-9 volt DC wall transformer. The 5V regulator is zesty enough to power both itself and an Arduino, so your Arduino-Propeller robot only needs one battery supply.
- Easy access to I/O pins 0 through 15 for prototyping.
- Six sets of three-pin headers for connecting servos or sensors.
- Three-channel analog-to-digital converter (ADC); two-channel digital-to-analog converter (DAC).
- Integrated stereo headphone audio amplifier for sound output, with standard 1/8" stereo jack.
- A µSD memory card slot.
- Various other features, including pin-selectable LEDs for visual indicators, standard VGA connector for video, and microphone.

As a sound co-processor, we'll use just a small portion of the solderless breadboard, the µSD card slot, and the stereo audio jack. The PropBOE uses a mini-USB connector for connecting to a PC. For programming, you'll need the Propeller Tool software, available from the Parallax website. See the **sidebar** for additional information on PropBOE downloads and resources.

Augmenting the Propeller with Objects

The Propeller is programmed using any of several languages. The most common is Spin, though the chip supports others including Basic, C, and a native assembly language called PASM. Spin is a self-descriptive language specially written to support the Propeller's unique functionality. As a programming language, Spin is not difficult to learn, but as an Arduino user you may find some of its syntax a tad exotic.

Right off the bat, you'll note that control structures — if, repeat, and so forth — are defined using indenting. This contrasts with the Arduino which uses brace characters to mark the start and end of structures. The indenting is useful for visualizing the structure of your programs.

The Spin language has all the routine programming statements you'd expect. For most everything else — setting up serial communications, running servos, or making sounds — the Propeller relies on object *files*, much in the same way that the Arduino uses object *libraries*. Several common objects come with the Propeller Tool, but others are available for download from various resources; see the **sidebar**.

Objects are just files, and by convention use the *.spin* file extension — also shared by regular Spin programs. Some objects involve multiple .spin files. All of the program examples for this project come with the required objects (except for the standard ones included with the Propeller Tool). You can download everything as a single zip file archive from the URL at the start of the article

The trick to using these extra objects is that they must reside in the same directory as the main Spin program. When you compile and upload your program to the Propeller, you only need to have the main program file open. The Propeller Tool will collect all the necessary elements and compile everything together.

A Brief Introduction to Sound

In order to better understand any sound generator, it's

Sources

Propeller Board of Education (PropBOE)
www.parallax.com/propboe

GN1:0 MIDI conversion tool (freeware)
www.gnmidi.com/gnfreeen.htm

Prerecorded sound clips
www.robotoid.com

WAV info at Wikipedia
http://en.wikipedia.org/wiki/WAV

Encoding software:
Audacity
audacity.sourceforge.net

Goldwave
www.goldwave.com

More Sound Generating Objects

Numerous sound-making objects have been created for the Propeller. Many are available on the Propeller OBEX (OBject EXchange) website, located at **obex.parallax.com**.

Sift through the Speech & Sound category. You'll find several handy sound generators, including:

SIDcog - Replicates an '80s style sound synthesizer chip, similar to the one on the Commodore 64.
Vocal Tract - Approximates the human voice for basic speech synthesis.
DTMF - Generates telephone dialing tones.

You can find additional objects on the Parallax forums. Use the forum search tool to find objects you're interested in.

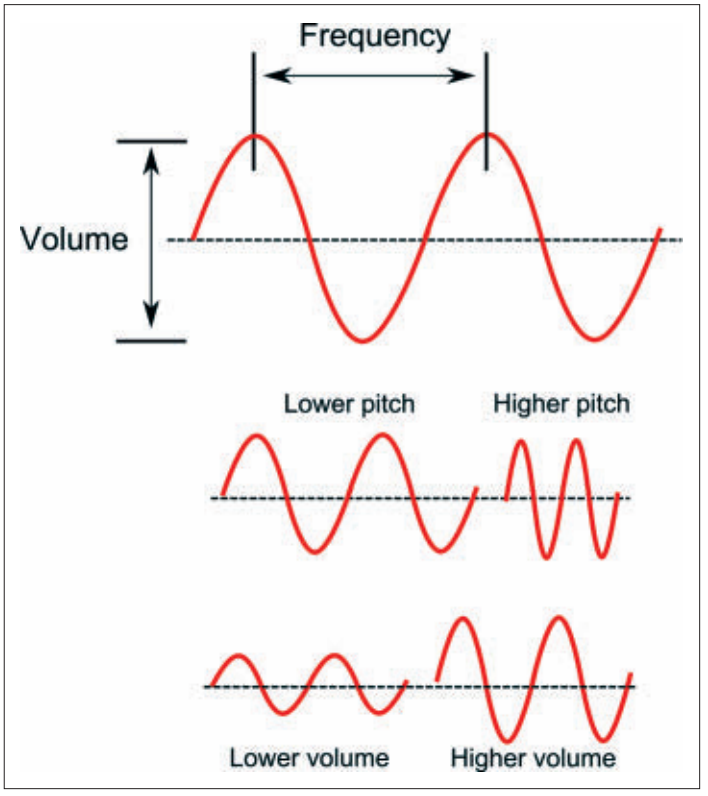


FIGURE 2. Sound waves and the electrical signals that produce them vary in frequency and intensity. The higher the frequency, the higher the pitch of the sound. Greater intensity produces louder volume.

handy to know a little about sound in general. So, here goes: Sound is waves that travel through some type of medium, like air or water. The size of the waves determines its amplitude — or volume — and how close the waves are from one another determines its frequency. The farther apart the waves, the lower the frequency and vice versa. Take a look at **Figure 2**.

When humans speak, a special muscle in the throat called the larynx vibrates at different speeds. Air from the lungs passes by the larynx, and the vibration of the muscle causes the air to become rapidly-changing sound waves. A similar process is involved in making sound from a guitar string (the string vibrates, disturbing the air around it) or a saxophone (a wooden reed vibrates as air is blown over it).

Electronic sound generators don't use a larynx, string, or reed, but they do use the same general principles of waves. Sound production begins with an oscillator circuit — in the case of a microcontroller, the oscillator is synthesized by using software to control a timer or counter. In its very basic form, the output of the oscillator is routed to a pin on the microcontroller which, in turn, is attached to a speaker. As the oscillator changes speed, the frequency of the tone heard through the speaker alters the pitch.

Most microcontrollers have more than one timer/counter, so it's technically possible to produce more than one tone at a time, with each tone at a different pitch. Each separate tone is called a *voice*. Any voice can be used by itself, though it is common to combine them to produce more elaborate sounds.

Sound is often characterized by the shape of the waves that produce it, not just the frequency of those waves. The shape of the wave is one thing that affects the *timbre* of the sound. Four common wave shapes include sine, saw tooth, triangle, and square, as depicted in **Figure 3**.

More sophisticated sound synthesis uses a concept called the envelope generator. It's a kind of dynamic volume control that varies over time. The repeating rising and falling sound of a siren is a simple example of an envelope. For more complex sounds — like synthesizing musical instruments — the envelope involves more elaborate and often rapidly occurring changes.

The basic sound envelope is characterized by four distinct phases (see **Figure 4**): attack, decay, sustain, and release, or ADSR. These phases are graphically shown as a series of ramps.

- *Attack* indicates how fast the sound comes to full volume, such as when a piano key is first pressed.
- *Decay* is how quickly the sound drops from its initial peak.
- *Sustain* is a constant volume of the sound after the decay once the key is let go.
- *Release* is how quickly the sound finally fades out.

These parameters (along with the wave shape) help to make the oscillation sound like something familiar to us — a piano or a violin, for instance. Each of these instruments has a peculiar ADSR envelope. Note that on some sounds, one or more of the ADSR phases may be very short or non-existent.

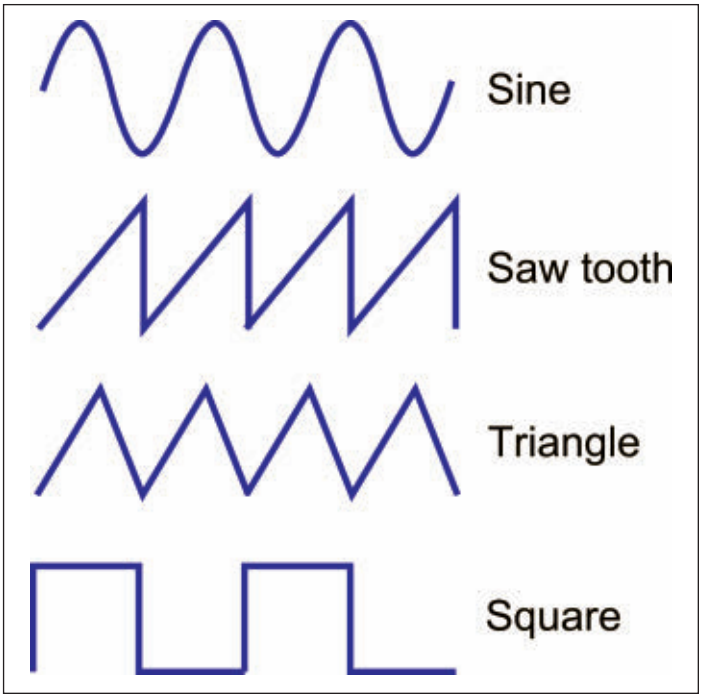


FIGURE 3. Four primary waveforms – sine, saw tooth, triangle, and square – produce different types of timbres that affect how the sound is perceived.

Using the Propeller to Generate Basic Tones

An easy task for the Propeller is to generate tones. The

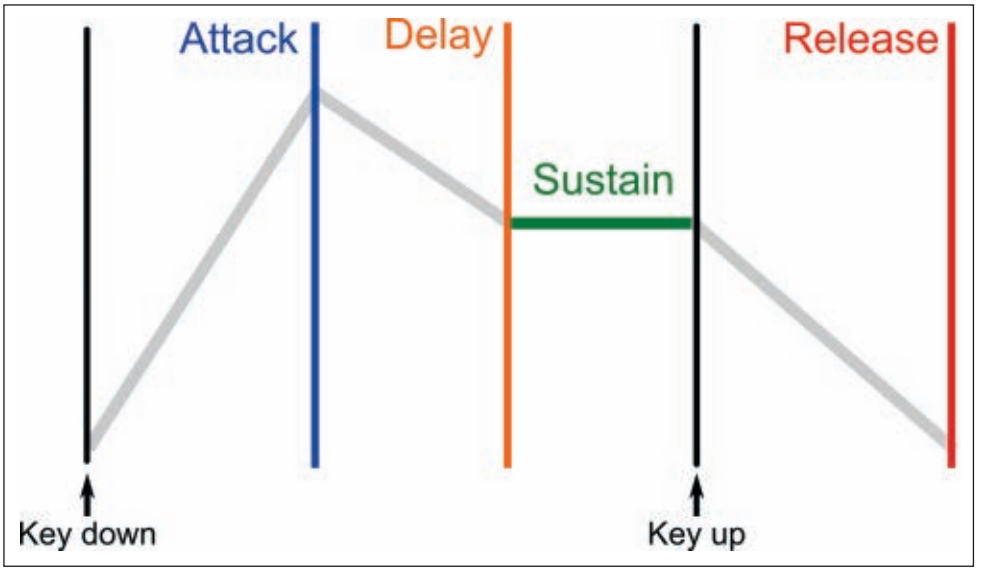


FIGURE 4. The four phases of the attack-delay-sustain-release (ADSR) envelope further define how humans perceive a musical instrument or other sound.

process involves using a cog counter (each cog supports two) to create a repeating pulse. The speed of the pulse defines the frequency or pitch of the tone. By turning the pulses on and off and altering the pitch, you can create basic tunes.

Listing 1 shows the main working part of a tone-making program for the PropBOE. The name of the program is *ToneMaker.spin*. (To save space, the program is not complete as shown; download the full version at the article link.) *ToneMaker.spin* repeats a series of seven C note tones a total of three times.

The layout of *ToneMaker.spin* is fairly typical of all Propeller Spin programs, and is worth reviewing:

- The OBJ section lists any objects required by the program. As noted above, these objects may either be included with the Propeller Tool, or must be in the same folder as the Spin program file. In the case of *ToneMaker.spin*, the program uses the FloatMath object which comes with the Propeller Tool software.
- The CON section lists any constants used in the program. Constants allow you to define such things as output pins and refer to them with descriptive names. Here, pin 27 is defined as the tone output. It's the left speaker connection on the PropBOE.
- A Spin program contains one or more PUB sections. The first PUB section — in this case, named *Go* — starts the main program.

Listing 2 (*ToneDemo.spin*) shows a variation on the tone-making theme. This program uses a separate object to handle tone generation. (The object file is not shown here; it's included in the file archive for this article.) The object also includes a handy electronic volume control — sound through the earphone jack on the PropBOE can be pretty loud!

ToneDemo.spin demonstrates the use of objects in a Spin program. In the OBJ section, the sound-producing object is given the unique reference name *spk*. You can use any name you'd like for object references, as long as it doesn't conflict with other objects or Spin commands. Then, each time you want to use the object you preface one of its methods with the *spk* reference name. For instance, to set the volume you use:

```
spk.speakerVolume
```

followed by the two parameters expected for this method: the desired volume (from 0 to 100), and the pin used as the output for the sound.

The file archive for this article contains an additional example (*NotePlayer.spin*) that's worth looking at. It mimics the Arduino tone statement, and contains 25 different canned tunes. Use it as a springboard for any sound project where you want your robot to play jingles.

Making MIDI Music on the Propeller

The tone generators discussed so far produce simplistic single-voice sounds. With creative use of its cog counters, the Propeller can synthesize fairly complex waveforms, approximating many types of musical instruments. Thanks to an open source MIDI object engine written by Propeller maven Andy Schenk, you can create fairly robust tones, sound effects, and even complete songs.

Listing 3 shows *MIDI-Chords.spin* — a demonstration program illustrating how to send MIDI commands to the MIDI engine object. The demo produces a series of C major

LISTING 1 — ToneMaker.spin

```
OBJ

  FloatMath  :  "FloatMath"     'Part of Propeller Tool

CON

  _clkmode = xtal1 + pll16x
  _xinfreq = 5_000_000
  OutPin = 27           ' Left speaker output

PUB Go

  repeat 3              ' Repeat 3 times
    Tone(33,   325)     'C1
    Tone(65,   500)     'C2
    Tone(131,  200)     'C3
    Tone(262,  750)     'C4
    Tone(523,  150)     'C5
    Tone(1047, 450)     'C6
    Tone(2093, 90)      'C7

    'Frequency, Duration
```

LISTING 2 — ToneDemo.spin

```
OBJ

  spk: "E555_SPKEngine.spin"

CON

  _clkmode = xtal1 + pll16x
  _xinfreq = 5_000_000

  _speakerPin = 27                  'Left speaker
  _speakerVolume = 75

  _spkMinFreq = 130
  _spkMaxFreq = 1300

PUB demo

  spk.speakerVolume(_speakerVolume, _speakerPin)

  repeat

    repeat result from _spkMinFreq to _spkMaxFreq step 100
      spk.speakerFrequency(result, _speakerPin)
      waitcnt((clkfreq >> 1) + cnt)

    repeat result from _spkMaxFreq to _spkMinFreq step 100
      spk.speakerFrequency(result, _speakerPin)
      waitcnt((clkfreq >> 1) + cnt)
```

chords using 10 of the object's pre-defined instruments. The *MIDI-Chords.spin* program shows turning the notes of the chords on, waiting for a brief period, then turning them off again. For more about MIDI commands, see Part 2 of this series.

The instruments built into the engine track the General MIDI 1 (GM1) set, comprising 127 instruments ranging from pianos to guitars to flutes. The synthesis may not be exact, but you can modify any sound in the MIDI engine by altering its characteristics. The MIDI object provides short documentation on how the synthesis works, and which parameters to change in order to alter the tonality of the instruments.

The PropBOE can also play MIDI tunes stored on a µSD card. The files must be in MIDI Format 0 (all one track). You can use a program like GN1:0 (see the **Sources** box) to convert the more common Format 1 files to Format 0. The file archive for this article contains some sample Format 0 MIDI files you can try. Look for the playlist section of the *MIDI-Player.spin* program. Add or remove the MIDI files you want to play (be sure the file uses standard 8+3 file naming conventions). Copy your files to a 1 GB or 2 GB µSD card that's formatted to FAT16.

Playing WAVes: About Sample Rate, Bit Depth, and Bit Rate

The PropBOE can directly reproduce sound and music from standard WAV digital audio files. We looked at MP3 audio last month, and in that discussion we discovered analog sound is converted to digital data by taking rapid samples of the sound at frequent and regular intervals. The number of times the sound is sampled in one second is the sampling rate. As a reference, music on a CD is sampled at 44,100 times a second (44.1 kHz).

Further, each sample is converted to a binary number. The number of bits in this number establishes the bit depth. The larger the depth, the higher the resolution — and quality — of the sound. Both eight- and 16-bit depths are common in WAV digital recordings.

The bit rate is the calculated product of the sample rate times the bit depth times the number of channels used for the recording. The bit rate is literally the number of bits processed per second. The higher the bit rate, the larger the encoded file will be. The formula for calculating the bit rate per second (bps) is:

*Sample Rate * Bit Depth * Channels*

So, given a 44.1 kHz sampling rate, bit depth of 16 bits, and two channels, the uncompressed bit rate is 1,411,200 bits per second — more commonly specified as 1.411 megabits per second (mbps).

The encoding format of this type of digital audio is referred to as linear pulse code modulation, or LPCM. The data is packaged into a structured file so that computer (and microcontroller) programs can readily access it. WAV (pronounced "wave") is among the most common file format for storing LPCM data. WAV files can store either uncompressed or compressed data, though uncompressed is far more common — and the kind we'll be using in this project.

Even a short four second WAV file is over 600 KB, which is far too large to load into the memory of a microcontroller. Instead, WAV file data is stored on SD or µSD solid-state memory which can hold many hundreds of audio clips.

Making WAV Audio Clips

You may already have some WAV clips you'd like to use for your robot. For this project, they need to be in a very specific format: 16-bit and two-channel (stereo). The sampling rate can be 22050, 32000, or 44100 — the higher the value, the larger the file. The file MUST NOT contain any metadata storing artist, genre, or other information.

Using an audio editing program like Audacity, you can convert existing clips to this format, or record and store new ones. You can trim extraneous parts from the clip, and apply effects such as echo, phasing, pitch, and speed. The process for converting or saving a WAV file in Audacity is pretty straightforward:

1. Open an existing sound clip. The higher the bit rate of the original, the better the output will sound. If prompted, choose the option "Make a copy of the files before editing."
2. Examine the Project Rate setting in the lower left corner. If it is not already, set it to 44100.
3. Look at the waveform in the center of the program window. Skip to Step 5 if it's already in stereo (you'll see two sets of waveforms). Otherwise, choose Edit->Duplicate.

This creates a duplicate waveform, adding it below the original.

4. On the top waveform, click the pull-down menu on the left and choose Left Channel. Do the same for the lower waveform, except choose Right Channel. (Note: This isn't true stereo, but rather binaural sound. In any case, it allows the resulting file to match the format criteria. As you gain experience with Audacity, you can synthesize a stereo sound by inserting a very slight delay to one of the channels, and adding phasing and other effects between the two channels.)

5. Choose File->Export. In the Export File dialog box, choose WAV (Microsoft) 16-bit signed PCM in the Save As type list.

6. Provide a name for the file and click Save. The file name must adhere to standard 8+3 name conventions. For convenience, use .wav as the extension; for example, *song0.wav* or *myfile.wav*.

7. Audacity will ask you to fill in numerous metadata fields, such as the artist name or genre. *Make sure all fields are empty!* Otherwise, the Propeller program will not be able to read the file.

LISTING 3 — MIDI-Chords.spin

```
CON

  _clkmode        = xtal1 + pll16x
  _xinfreq        = 5_000_000
  Delay           = 80_000_000

  Right           = 26
  Left            = 27

OBJ

  synth : "pm_synth_20"

VAR

  long note
  long instrument
  long channel
  long velocity

PUB Main

  synth.start(Left,Right,2)               ' Start synth with 20 voices

  note := 60                              ' Middle C
  instrument := 33                        ' Start with instrument 33
  channel := 0                            ' User channel 0
  velocity := 127                         ' Set volume (127=loudest)

  ' Play major chords
  repeat 10                               ' Cycle through 10 instruments
    synth.prgChange(instrument, channel)  ' Set instrument

    synth.noteOn(note, channel, velocity) ' Pitch, channel, velocity
    synth.noteOn(note+4, channel, velocity)    ' Major third
    synth.noteOn(note+7, channel, velocity)    ' Perfect fifth
    waitcnt(Delay+cnt)

    synth.noteOff(note, channel)          ' Turn off all notes
    synth.noteOff(note+4, channel)
    synth.noteOff(note+7, channel)
    waitcnt(Delay*2+cnt)

    Instrument+=3                         ' Skip every 3 instruments

  synth.allOff                            ' All notes off
  synth.stop                              ' Turn off MIDI when done
```

If the clip you want to export doesn't exist, you can create it directly in Audacity. Connect a microphone to your PC, and choose 44100 for the Project Rate. Click on the Record button, then record the clip. When you're done, click the Stop button to terminate recording. Follow Steps 5 through 7 above to save the file to disk.

Reading and Playing a WAV File Using the PropBOE

With your sound file(s) complete, transfer them to a name brand (SanDisk, Sony, Kensington, etc.) µSD card. The card must be formatted for FAT16. For now, avoid cards over 2 GB, and keep away from "bargain" SD cards because they may be out of spec and cause errors. Copy all files to the root directory of the card. (Note that the PropBOE can use larger cards formatted for FAT32, but the SD objects included with this month's projects are simpler versions that only support FAT16.)

To read a WAV file on a memory card, the PropBOE must:

1. Open (or mount) the SD card. This allows the files on it to be read.
2. Locate the file you want to play and scan its header to retrieve metadata that indicates how the file was encoded. Critical details include the sample rate (e.g., 44.1 kHz) and the number of samples in the file.
3. Read off chunks of data from the file. Each chunk is then processed by sending it through a DAC circuit. This converts the digital data to audible sound.
4. Repeat the process until the entire file has been read.
5. Close (unmount) the SD card.

The PropBOE contains all the hardware needed to read a µSD card and process WAV files on it. A built-in audio amplifier lets you connect a pair of headphones directly to the board. Or, you can plug in an external stereo amp and speaker — a small PC speaker system is ideal.

LISTING 4 — PropellerTestPlay.spin

```
OBJ
     SD        :    "FSRW"
     debug     :    "FullDuplexSerial"

CON _clkmode = xtal1 + pll16x
     _xinfreq = 5_000_000

     SD_basepin = 22         ' Base (first) pin for SD card
     RPin = 26               ' Right channel pin
     LPin = 27               ' Left channel pin

     ' Set pins and baud rate for Serial Terminal window
     debug_Rx = 31
     debug_Tx = 30
     debug_Baud = 115200

     buffSize = 100

VAR long parameter1          ' @buff1 to ASM
     long parameter2         ' @buff2 to ASM
     long parameter3         ' sample rate to ASM
     long parameter4         ' #samples to ASM
     long buff1[buffSize]
     long buff2[buffSize]
     byte Header[44]

PUB Main | n,i,j, SampleRate,Samples, cmd, WAVCOG

   ' Initialize comm port for monitor window
   debug.start(debug_Rx, debug_Tx, 0, debug_Baud)


   debug.str(string("Starting Up", 13))
   ' Open the WAV file (only 16-bit PCM WAV Files) on SD card
   i:=sd.mount(SD_basepin)
   if (i<>0)
     repeat

   i:=sd.popen(string("test0.wav"), "r")   ' -- Play file --

' See full program in archive file for remainder
```

Listing 4 (*PropellerTestPlay.spin*) provides a basic test bench for checking the WAV playing operation. It's based on a series of handy WAV player examples written by Spinmeister Ray Allen, and is available on the Propeller OBEX website. First, create or convert a short (10 seconds will do) WAV clip, name it *test0.wav*, and copy it to a suitable µSD card. As a reminder, the file, µSD card, and WAV file must conform to the following:

- Name brand, 1 GB or 2 GB µSD, formatted to FAT16.
- WAV file encoded at 16-bits, stereo. The sample rate can be your choice, but should not be greater than 44100 bits per second.

Upload the *PropellerTestPlay.spin* program to the Propeller BOE. Note that **Listing 4** shows only the top portion of the *PropellerTestPlay.spin* program. The remainder contains assembly language code that is difficult to accurately retype. This full program is in the article archive.

You should hear the WAV file played through the Propeller BOE and headphones or amplifier/speaker. If you don't hear anything, double-check connections, and be sure the µSD card is securely inserted into its slot. The *PropellerTestPlay.spin* program provides rudimentary debugging information. You can view it by opening up the Parallax Serial Terminal program which comes with the Propeller Tool download (press F12 when in the Propeller Tool). See the Propeller Tool instructions for how to set up and use the Serial Terminal.

Still no playback? Ensure your files conform to 8+3 naming — *test0.wav* is good, but *myfirsttest.wav* is not. Try encoding at a lower sampling rate, either 22050 or 32000 bits per second. Be *absolutely sure* no artist, genre, or other metadata is being stored in the file (if so, resave from scratch). Also, try opening the file in a different sound editing program, then resaving at the same or different sampling.

Controlling WAV Playback From the Arduino

With a bit of extra coding, all of the sound generation methods detailed so far can be controlled from an external source. Using an Arduino as a host controller, you can send simple commands to the PropBOE via a serial link. The Arduino sends a short coded message; the Propeller, in turn, receives the message and plays the corresponding sound.

Listing 5 shows a basic Arduino sketch for demonstrating this technique. Connect three momentary pushbutton switches to the Arduino as illustrated in **Figure 5**. Pressing switch S1 or S2 triggers one of two WAV files stored on the µSD card. (The audio files are named *test0.wav* and *test1.wav*. Each should be fairly short — between five and 10 seconds in length.) Switch S3 serves to immediately cancel playback, if sounding.

Important! Don't forget the 1K series resistor between the Arduino and Propeller. The standard Arduino (such as the Uno board) is a 5V device; the Propeller is a 3.3V device. The series resistor reduces the current flowing over the connecting wire.

Pressing button S1 or S2 sends a three-byte sequence that starts with the ! (exclamation point) character, followed by W (for WAV), and then a number character indicating the file to play. On the Propeller end, the characters are intercepted and parsed to determine if the message is something of interest. Only messages that begin with ! are processed. The ! serves to keep the communications between the Arduino and Propeller in sync.

Finally, pressing switch S3 causes the Arduino to transmit the sequence !XX. The ! character is again used to maintain synchronization, and the X means cancel. The parsing mechanism in the Propeller program is rudimentary, and all three messages have the same three-character length to simplify coding.

The companion Propeller Spin program — *PropellerParseIn.spin* — is too lengthy to show here, but it's

included in the article archive. This program reads the serial port created on pin P0 of the PropBOE, waiting for the !W code sequence to come from the Arduino. Once these characters are received, the program then checks the last character of the code to determine which WAV file to play.

Two cogs are created and later removed every time an audio clip is sounded. Each cog runs a particular piece of code in the program:

- A separate cog executes a series of Assembly language statements (bottom of program) that do the actual data fetching from the µSD card. Assembly language is used here for its speed of execution.
- Once a WAV file starts to play, another cog listens for data arriving on the software serial port connected to the Arduino. This serial port is defined on pin P0 of the PropBOE. The cog contains a small background method that constantly scans the serial port looking for the !XX cancel code.

Both cogs are terminated after the WAV file has played, or if playing has been canceled from the Arduino. The program then goes back to its idling state, waiting for the next !W code sequence.

Adapting the Co-process Control to Other Sound Services

You can utilize the same techniques used in the *PropellerParseIn.spin* program to control other sound-making processes. You can change the coding to trigger a different type of sound. For example, the prefix !M might trigger a MIDI song. The third character is the specific song to play. You can use any byte value you like, except value 33 — that's the ! character.

The example *SerialOut.ino* sketch sends the number characters 0 and 1; you can also use letters and most other characters to increase the available selection. On the Arduino side, send the code characters in one group like this:

```
SerOut.print("!MA");
```

On the Propeller side, validate against the expected third character as so:

```
case cmd
  "A":
```

Other variations might include selecting from a set of canned tunes using tone generation, speaking short phrases using a voice synthesizer object, or sound effects using the MIDI engine.

With your Propeller sound co-processor set to make noise, you can now integrate it with your robot. Add a pair of leaf switches to the front of your bot like we did in Part 3 of this series. If your robot touches an obstacle, the switches trigger a short song that plays while the bot backs up and heads off in a new direction.

The more you use sound with your robots, the more you'll come to rely on it for providing feedback, warnings, and even entertainment. With the benefit of a sound co-processor built on the Propeller, the functionality of your robot doesn't have to suffer because it's busy playing a tune or blasting its "move out of the way!" siren. SV

LISTING 5 — SerialOut.ino

```
#include <SoftwareSerial.h>
SoftwareSerial SerOut(255, 8);

void setup() {
  SerOut.begin(9600);

  pinMode(5, INPUT);
  pinMode(6, INPUT);
  pinMode(7, INPUT);
  digitalWrite(5, HIGH);
  digitalWrite(6, HIGH);
  digitalWrite(7, HIGH);
}

void loop() {

  // Clip #0
  if(digitalRead(5) == LOW) {
    SerOut.print("!W0");
    delay(500);
  }

  // Clip #1
  if(digitalRead(6) == LOW) {
    SerOut.print("!W1");
    delay(500);
  }

  // Cancel play
  if(digitalRead(7) == LOW) {
    SerOut.print("!XX");
    delay(500);
  }
}
```

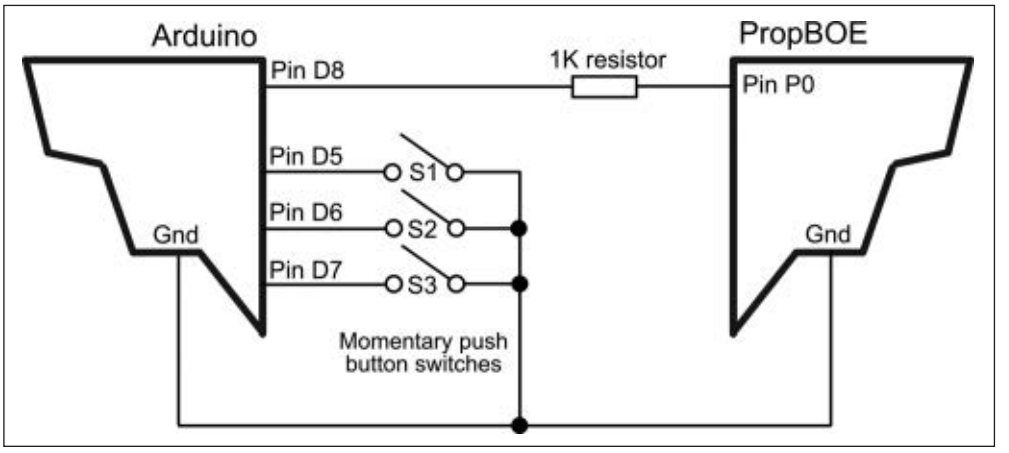


FIGURE 5. Connection diagram for communicating between an Arduino and PropBOE. Pin D8 from the Arduino is the serial output and connects to pin P0 of the PropBOE. Momentarily depressing one of the buttons on the Arduino causes it to transmit a short serial data code to the Propeller.

The SERVO Webstore

Attention Subscribers ask about your discount on prices marked with an *

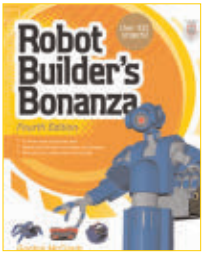
CD-ROM SPECIALS

NEW RELEASE

ROBOTICS

Robot Builder's Bonanza, Fourth Edition

by Gordon McComb



Robot Builder's Bonanza, Fourth Edition includes step-by-step plans for a number of motorized platforms. The book is described as a compendium of robotics topics, containing more than 100 projects, including 10 robot designs new to the fourth edition. These modular robots are low cost, and are made to be reproduced by readers with no training in mechanical construction.

$29.95*

Mechanisms and Mechanical Devices Sourcebook 5th Edition

NEW!

by Neil Sclater

Fully revised throughout, this abundantly illustrated reference describes proven mechanisms and mechanical devices. Each illustration represents a design concept that can easily be recycled for use in new or modified mechanical, electromechanical, or mechatronic products. Tutorials on the basics of mechanisms and motion control systems introduce you to those subjects or act as a refresher.

Reg $89.95 Sale Price $79.95

Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists

by Dustyn Roberts

In *Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists*, you'll learn how to successfully build moving mechanisms through non-technical explanations, examples, and do-it-yourself projects — from kinetic art installations to creative toys to energy-harvesting devices. Photographs, illustrations, screenshots, and images of 3D models are included for each project.

$29.95*

Build Your Own Humanoid Robots

by Karl Williams

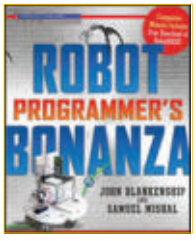
GREAT 'DROIDS, INDEED!

This unique guide to sophisticated robotics projects brings humanoid robot construction home to the hobbyist. Written by a well-known figure in the robotics community, *Build Your Own Humanoid Robots* provides step-by-step directions for six exciting projects, each costing less than $300. Together, they form the essential ingredients for making your own humanoid robot.

$24.95*

Robot Programmer's Bonanza

by John Blankenship, Samuel Mishal



The first hands-on programming guide for today's robot hobbyist!

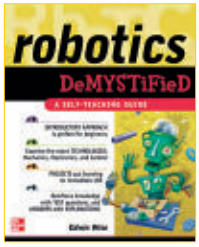
Get ready to reach into your programming toolbox and control a robot like never before! *Robot Programmer's Bonanza* is the one-stop guide for everyone from robot novices to advanced hobbyists who are ready to go beyond just building robots and start programming them to perform useful tasks.

$29.95

Robotics Demystified

by Edwin Wise

YOU DON'T NEED ARTIFICIAL INTELLIGENCE TO LEARN ROBOTICS!



Now anyone with an interest in robotics can gain a deeper understanding — without formal training, unlimited time, or a genius IQ. In *Robotics Demystified*, expert robot builder and author Edwin Wise provides an effective and totally painless way to learn about the technologies used to build robots! **$19.95**

We accept VISA, MC, AMEX, and DISCOVER
Prices do not include shipping and may be subject to change.

To order call 1-800-783-4624

SERVO Magazine Bundles

Save $10.00 Only $57.95!

Now you can get one year's worth of all your favorite articles from *SERVO Magazine* in a convenient bundle of print copies. Available for years 04, 05, 06, 07, 08, 09, 10 and 2011.

Any bot builders out there? Get cool robotics stuff from my store!

Call me at my order desk!

Visit my online store @ www.servomagazine.com

RobotBASIC Projects For Beginners

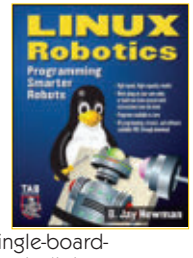
by John Blankenship, Samuel Mishal

If you want to learn how to program, this is the book for you. Most texts on programming offer dry, boring examples that are difficult to follow. In this book, a wide variety of interesting and relevant subjects are explored using a problem-solving methodology that develops logical thinking skills while making learning fun. RobotBASIC is an easy-to-use computer language available for any Windows-based PC and is used throughout the text.

Price $14.95

Linux Robotics

by D. Jay Newman

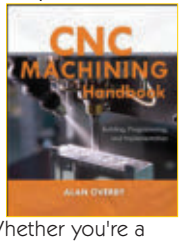


If you want your robot to have more brains than microcontrollers can deliver — if you want a truly intelligent, high-capability robot — everything you need is right here. *Linux Robotics* gives you step-by-step directions for "Zeppo," a super-smart, single-board-powered robot that can be built by any hobbyist. You also get complete instructions for incorporating Linux single boards into your own unique robotic designs. No programming experience is required. This book includes access to all the downloadable programs you need.

$34.95

CNC Machining Handbook: Building, Programming, and Implementation

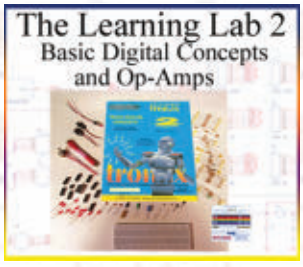
by Alan Overby



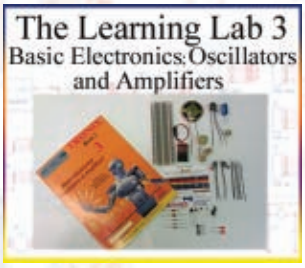
The *CNC Machining Handbook* describes the steps involved in building a CNC machine and successfully implementing it in a real world application. Helpful photos and illustrations are featured throughout. Whether you're a student, hobbyist, or business owner looking to move from a manual manufacturing process to the accuracy and repeatability of what CNC has to offer, you'll benefit from the in-depth information in this comprehensive resource. **$34.95**

FOR BEGINNER BOT BUILDERS

$59.95



$49.95



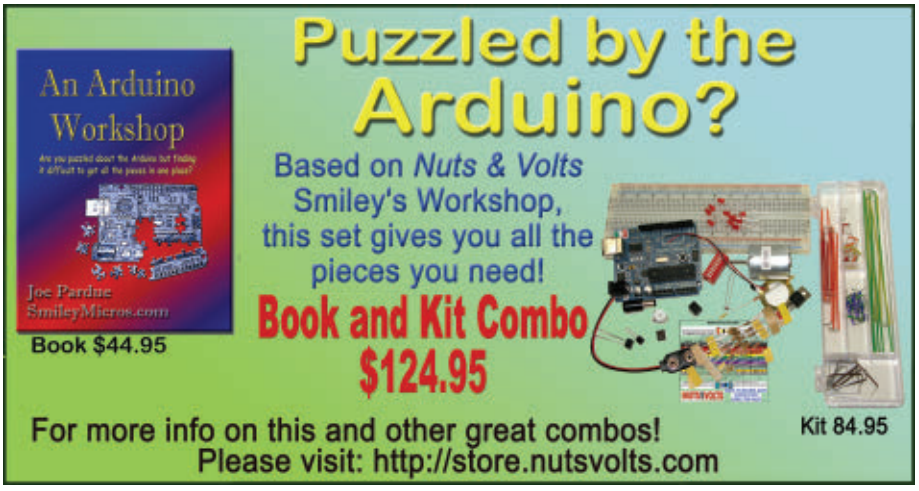
$39.95

The labs in this series — from GSS Tech Ed — show simple and interesting experiments and lessons, all done on a solderless circuit board. As you do each experiment, you learn how basic components work in a circuit, and continue to build your arsenal of knowledge with each successive experiment.

For more info and a promotional video, please visit our webstore.

Or order online
www.servomagazine.com

SPECIAL OFFERS



Forbidden LEGO

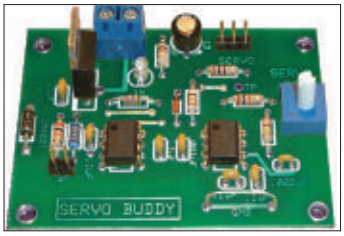
by Ulrik Pilegaard / Mike Dooley

Forbidden LEGO introduces you to the type of free-style building that LEGO's master builders do for fun in the back room. Using LEGO bricks in combination with common household materials (from rubber bands and glue to plastic spoons and ping-pong balls) along with some very unorthodox building techniques, you'll learn to create working models that LEGO would never endorse.

Reg $24.95 Sale Price $19.95

PROJECTS

The SERVO Buddy Kit



An inexpensive circuit you can build to control a servo without a microcontroller.

For more information, please check out the May 2008 issue or go to the SERVO webstore.

Includes an article reprint.
Subscriber's Price **$39.55**
Non-Subscriber's Price **$43.95**

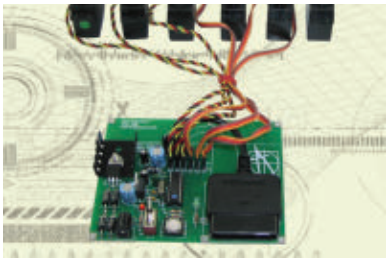
3D LED Cube Kit

From the article "Build the 3D LED Matrix Cube" as seen in the August 2011 issue of *Nuts & Volts Magazine*.

This kit shows you how to build a really cool 3D cube with a 4 x 4 x 4 monochromatic LED matrix which has a total of 64 LEDs. The preprogrammed microcontroller that includes 29 patterns that will automatically play with a runtime of approximately 6-1/2 minutes.
Colors available: Green, Red, Yellow & Blue.
Jig and plastic cases also available.

Subscriber's Price **$57.95**
Non-Subscriber's Price **$59.95**

PS2 Servomotor Controller Kit



This kit accompanied with your own PlayStation controller will allow you to control up to six servomotors. Includes all components and instruction manual.

For more information, please see the February 2011 edition of *SERVO Magazine*.
Assembled units available!
Subscriber's Price
$79.95
Non-Subscriber's Price
$84.95

Building Maxwell: The Software

Part 3

by Michael Ferguson

Last month, we looked at the mechanical and electrical construction of a low-cost mobile manipulator named Maxwell. This article will focus on the software used; namely the various components of the Robot Operating System.

ROS Basics

The Robot Operating System — or ROS — is an open source framework that aims to aid developers in making robots more useful. In just three short years of development, the ROS ecosystem has grown to thousands of developers with over 100 different university labs, companies, and individuals releasing open source software targeted at ROS. While a first glance at the ROS wiki can appear daunting, the good news is there are relatively few concepts in ROS.

ROS takes a distributed approach. Rather than having all of your code wrapped up into a single executable, a typical robot will have many separate programs (called nodes) — all communicating with one another by passing messages over named topics. As long as both ends of the communication can decode the message, the two programs really don't even have to know they are separate programs (or even that they may be running on different computers). ROS provides client libraries which allow nodes to connect to topics and decode messages in several popular languages, although the best support is available for C++ and Python.

To facilitate the easy distribution of code, software is wrapped up into collections of

FIGURE 1. Maxwell's URDF visualized in RVIZ.

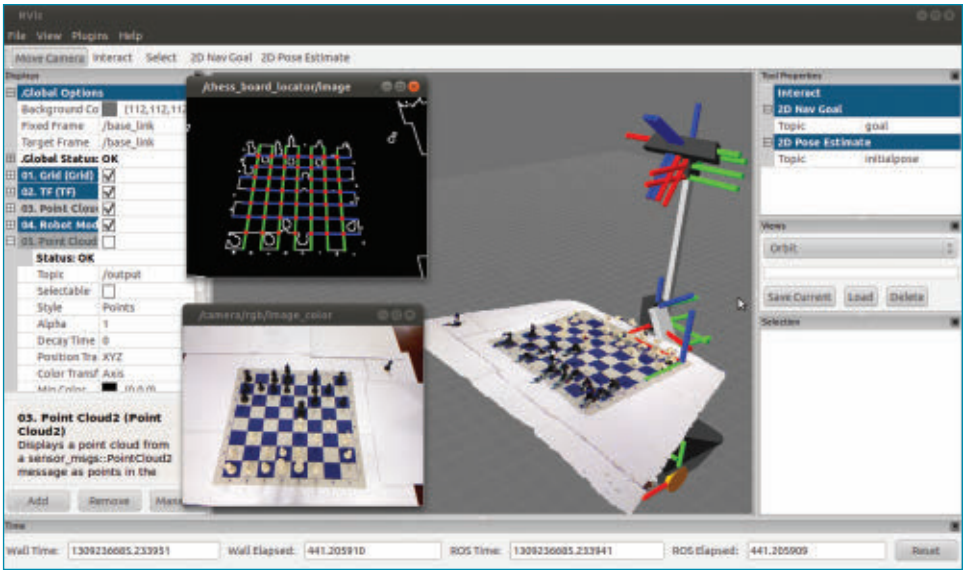


FIGURE 2. Merging sensor data in RVIZ: A view of Maxwell recognizing a chess board.

However, as the rest of this article will point out, once you have gotten past the initial learning curve, the returns can be great. ROS has an excellent set of tutorials which can be found at **www.ros.org/wiki/ROS/Tutorials**.

Hardware Drivers

There are ROS bindings to hardware drivers for a large number of robotic platforms. A search for 'drivers' on the list of ROS software returned over 100 existing packages. This article will only look at the drivers used for Maxwell. All of the custom source code and configuration packages used for Maxwell are hosted at **http://code.google.com/p/vanadium-ros-pkg**.

Most of Maxwell's hardware is connected to the PC through the ArbotiX board, for which I wrote the ROS drivers. The ArbotiX drivers are highly reconfigurable, but we do have to set up a configuration file which maps Dynamixel servo IDs to real world names.

In addition to the ArbotiX, Maxwell uses an Asus Xtion RGBD camera which uses the same openni_kinect drivers as the Kinect.

At this point, you may start to think that one would have to start all these programs individually every time a robot is started up, which could become very tedious. Luckily, ROS has an XML-based 'launch' file which can be used to configure and start numerous ROS nodes.

All of the configuration and launch files are stored in the maxwell_defs package which also contains numerous other hardware definitions described in the next section. When starting up Maxwell, only one launch file is needed: bringup.launch. Using tools such as Upstart, you can even have that start automatically if the computer is dedicated to your robot.

packages where an individual package might offer a few related executables or configuration files. Related packages are then wrapped up into stacks. When installing parts of ROS, the general work flow is to install a number of stacks. If using Ubuntu Linux, many of the better supported stacks can be installed from debs using the apt-get tool.There is a single shared wiki at **www.ros.org**, but developers set up their own source hosting; many host on Google code.

Since different nodes in ROS communicate through messages, a number of standard messages have arisen for common use cases. For instance, when controlling a mobile base, it is almost universally true that the hardware drivers will accept commands in the form of a geometry_msgs/Twist as the commanded velocity; geometry_msgs is a package containing message definitions for various geometric notations, and a Twist is a velocity of a rigid body. At the end of the day, any robot that takes a Twist command can interact with any program that outputs a Twist command (with possibly some tuning for speed/acceleration limitations of different hardware).

Development on ROS is on-going. It is currently targeted primarily at Linux, and there is a learning curve.

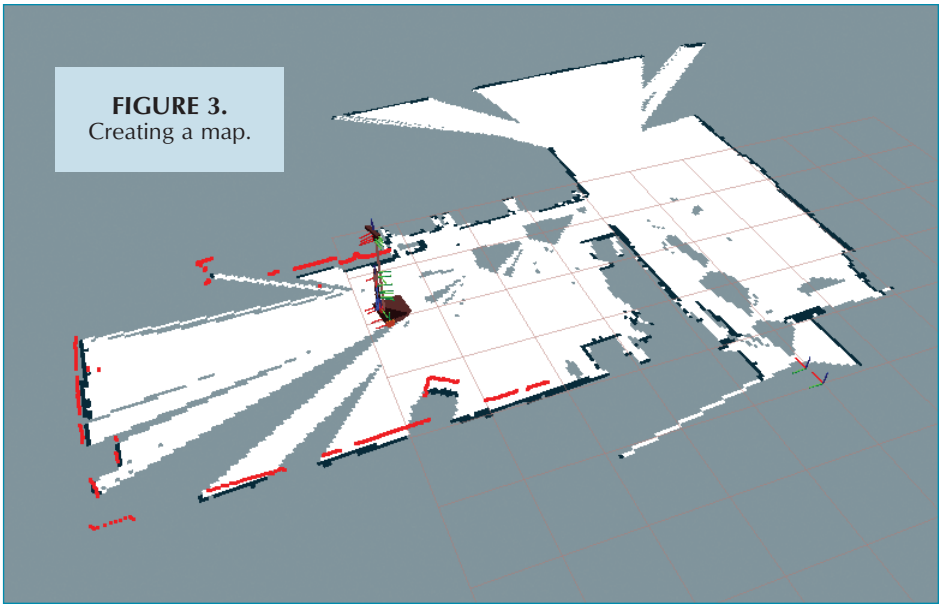


FIGURE 3. Creating a map.

The URDF and TF

TF and the URDF are two of the most useful features in ROS (next to all those other useful features). URDF is the Universal Robot Description Format. Basically, it is an XML-based file format that lets you describe how the joints and links of robots come together, how links should be visualized, and how collision planning should regard links (often, you want to use a lower resolution mesh for collision planning than for visualization). Thus, when putting together a new robot in ROS, one of the first things you often do is create a URDF for the robot. Obviously, this makes clear sense if you want to do any sort of planning that needs collision data, but what other reason do you have to create URDF?

This is where TF comes in. TF is the transform system in ROS which allows transform data to flow seamlessly through your network of ROS nodes. When you create URDF and have a hardware driver which is broadcasting the state of individual joints, TF will allow you to visualize the robot state, or translate sensor data between connected frames. So, if your robot has a Kinect on a pan/tilt head and sees a Coke can on a table, TF will let you convert the camera coordinates of that can into a different frame — possibly the arm's base_link or the base of your mobile manipulator — if you want to run over the can. With a URDF and TF, you no longer have to spend weeks rewriting (and debugging) the same mathematical transformations over and over again each time you create a new robot or app.

As a starting point towards creating your own URDF, many of the popular Robotis servos and brackets are already available in Maxwell's software repository.

Visualization: RVIZ

One of the most important parts of robot development is the quality of your debugging tools. ROS offers a number of debugging tools including command line utilities that query the status of the ROS network, small GUI tools which let you plot numeric messages, and the 3D visualized RVIZ. RVIZ actually lets you merge all sorts of data together in a single 3D viewer; each message gets transformed into the viewer's frame using TF. With RVIZ, you can actually see what your robot is seeing and thinking, which really helps when it comes to debugging.

Base Navigation

The navigation stack is probably one of the most used robot apps available in ROS. While it used to take months or years of coding to get a robot to autonomously navigate a space, now it is simply a matter of tuning parameters. While a NAV stack used to require an expensive scanning laser range finder, the TurtleBot developers have released a set of tools that can convert the Kinect data into a fake laser scan.

The navigation stack is actually several nodes working together to get a robot navigating. One node (AMCL) handles localization of the robot within a map, while a separate node containing a local and a global planner tries to send goals to the mobile base which are collision free and working towards a goal. Of course, even the best sensors sometimes fail, so the navigation stack also includes a number of recovery behaviors.

The navigation stack has a number of tutorials on usage and setting it up for your ROS-enabled robot at **www.ros.org/wiki/navigation**.

Arm Navigation

The arm_navigation stack follows the same general idea of the navigation stack, but is aimed at the 3D navigation of high degree-of-freedom arms. Arm navigation provides several possible tools for creating 3D occupancy grids, a number of state-of-the-art planners, and trajectory filters to smooth out the arm motion.

While it may seem that arm navigation would actually be more difficult to set up since high degree-of-freedom arms are more complex than mobile bases, it is actually easier since the arm_navigation team has put together a nifty Wizard that steps through the setup. However, arm_navigation very much needs a real URDF with collision models for all links in the arm or it cannot actually plan. Tutorials are available at **www.ros.org/wiki/arm_navigation**.

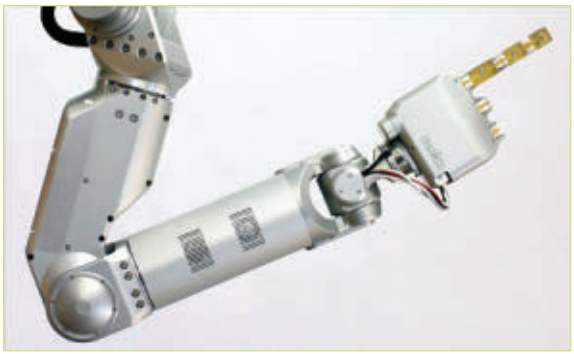
Conclusion

This series of articles covered some of the basics of building a mobile manipulator using Dynamixel servos, the ArbotiX RoboController, and ROS. I hope you'll follow future development of Maxwell on my blog, which can be found at **http://showusyoursensors.com**. SV

www.servomagazine.com/index.php?/magazine/article/June2012_Ferguson

Discuss this article in the *SERVO Magazine* forums at http://forum.servomagazine.com

Continued from page 31



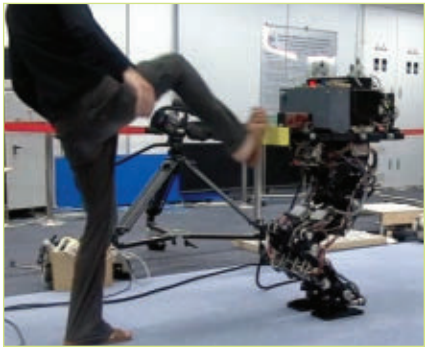
AN ARM UP ON COMPETITION

Aaron Edsinger of Meka Robotics, has announced an entirely new company called Redwood Robotics — a joint venture between Meka Robotics, Willow Garage, and SRI International. While Redwood has apparently been in the works for about a year, Edsinger wasn't ready to commit to much as far as what the new company will actually be working on. However, here's what we know:

Redwood will be building the "next generation arm" for robots. Edsinger wants to create an arm that does for robotics what the Apple II did for computers: Get the hardware out of factories and into homes. In other words, something that's simple to program, inexpensive, and safe to operate alongside people. Specifically, they'll be working on a "new type of device" with "new types of interfaces," providing "product solutions for integrators, developers, and enterprise customers" who (we'll assume) will then sell the arms with software or firmware to end users. Long term, Redwood wants to be the "arm merchant for emerging personal and service robot markets." So, when you buy your robot butler in a few years, it'll be from some company that isn't Redwood, but Redwood hopes it will supply that robot's arms to the company you do buy it from.

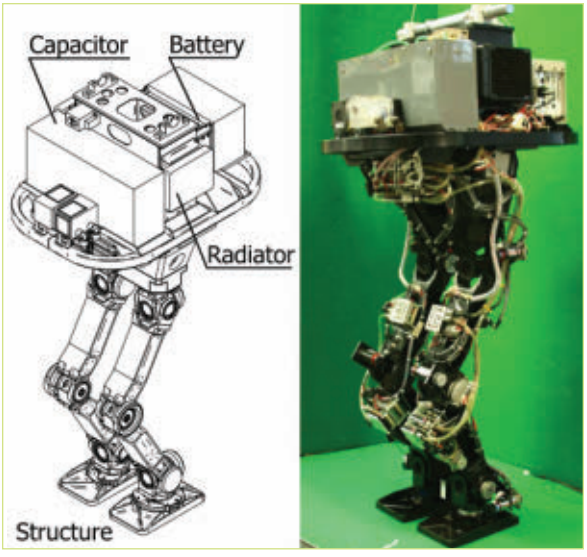
Redwood Robotics isn't the only company that's working on low cost robot arms. The other big name in this space is Heartland Robotics, a stealthy venture-backed startup founded by Rodney Brooks. Last we heard (Dec '10), Heartland was working on a human-safe robot for industrial assembly and packaging that reportedly would cost less than US $5,000.

Visitors to Heartland describe a robot that looks like a human from the waist up, with a torso, either one or two arms with grippers, and a camera where you might expect the head to be. The robot is on a rolling base rather than legs. It can be moved around but doesn't move autonomously. The arm and gripper can be quickly trained to do a repetitive task just by moving them — no software code required. Heartland targets industrial markets, while Redwood seems to be focused on personal and service robotics.



WHAT A KICK!

For some reason, roboticists seem to enjoy testing their creations by kicking them, punching them, shoving them, and even striking them with baseball bats and heavy pendulums. (All in the name of science, of course.) If we want robots that can do chores around the house, care for the elderly, or (if you're a DARPA program manager) drive trucks and crash through walls, then we need robots with actuators that are both fast and strong. The problem is actuators based on electrical motors can only deliver a limited amount of power, and the alternative — hydraulics — requires bulky pumps and can be difficult to control. Junichi Urata and his colleagues at the University of Tokyo's JSK Lab, led by Professor Masayuki Inaba, are working on a possible solution. They've developed a high torque, high speed robotic leg based on a novel electrical actuation system. Their robot uses high voltage and high current liquid-cooled motor drivers that get their power from a 13.5 farad capacitor system. Why a capacitor? Because it can supply lots of current very fast and reliably — something that batteries are not good at. The researchers modified an existing HRP3L (developed by Kawada Industries) to create their robot which they call HRP3L-JSK. Thanks to the capacitor-powered motor drivers, the robot's Maxon 200 watt brushless motors (modified to be liquid-cooled) can achieve instantaneous speeds of over 1,000 degrees per second and 350 Nm of torque on the robot's knee joint. This capability allows the 53 kg robot to react to disturbances (in this case, kicks, knee strikes, and other abuse from researchers) and even jump 44 centimeters off the ground (though the landing part will need work).



The robot relies on a new balance control system that detects disturbances and computes 170 foot placement possibilities in one millisecond, choosing the best candidate to keep the robot from falling. The new method is a collaboration between the JSK team and researchers from Japan's National Institute of Advanced Industrial Science and Technology (AIST).

Urata, who recently received a Ph.D. degree for his HRP3L-JSK work, now has his eyes on the DARPA Robotics Challenge.

QBO FOR YOU-BO?

Qbo, from TheCorpora, is essentially a bunch of well-designed hardware that's intended to take all of the making a robot out of making a robot. It's similar to the philosophy behind other robot kits: If you buy a Qbo, you don't have to worry about spending a lot of time and money building a robot from scratch that can do what Qbo can do.

So, what can Qbo do? Well, that depends on what you want it to do which, in turn, depends on what version you want to get and how much you want to spend. Qbo will be available in three different flavors: Basic, Lite, and Pro. Here are the differences:

• Qbo Basic: If you want to customize all of Qbo's guts, Qbo Basic is just a chassis with all the plastic covers and mechanical parts, along with HD webcams and a set of controller boards. There are no servos, power systems, cables, computers, or other sensors, giving you the freedom to trick your Qbo out just the way you want. You'll need to do a bunch of work yourself, and there's a lot of additional hardware you'll need to buy, but a lot of that stuff will more or less just drop right in.

• Qbo Lite: For researchers who want a robot that works out of the box (like all you software types), Qbo Lite includes sensors and computers, and comes fully assembled and ready to go. The hardware isn't particularly fancy (an Intel Atom processor, 2 gigs of RAM, and a 40 gig mechanical HD), but you do get some range finders and proximity sensors, and a nice little LCD and those HD webcams. In any case, it'll certainly get the job done and allow you to start playing around immediately.

• Qbo Pro: The difference between Qbo Lite and Pro is that the Pro version comes with substantially beefed-up hardware. Along with more powerful actuators, Qbo gets an Intel Core i3 processor and a 40 gig SSD.

As far as the software goes, Qbo runs Linux and ROS, and also comes with access to the OpenQbo open source community which provides a place where people can share Qbo-specific hardware and software. Current applications include 3D vision, speech recognition, voice synthesis, face recognition, object recognition, telepresence, and SLAM.

Then and NOW

Sensors For Mobile Robots — Part 2
Location and Object Recognition

by Tom Carroll

Last month, I began by referencing Bart Everett's sensor book from 1995, *Sensors for Mobile Robots* — a book that, in my opinion, should be on every robot experimenter's bookshelf. Everett's book emphasizes just how critical the need and use of sensors is for successful robot designs. His book not only describes how they work, but which ones to use and how to use them. Though it was published 17 years ago, almost all of the types and technology involved are still appropriate for today's robot designs.

In May's article, I highlighted Bart's Robart I and Robart II as examples of some pretty sophisticated robots. His Robart III — begun in 1992 and a work-in-progress at the time his book was published — is shown in **Figure 1**. Bart's third-generation robot sported some unique 'military appendages' such as a pneumatically powered six-barrel Gatling-style gun as its right arm shown in the **figure**. It fires simulated tranquilizer darts or rubber bullets, strictly for demonstration purposes. Down at the base, you can see the blue SICK LIDAR ranging system that replaced some of the original Polaroid ultrasonic distance ranging sensors, though nine remain on the lower base, the lower waist, and the arms. Robart III — like its predecessors — is strictly a laboratory prototype and certainly could not be used on a modern battlefield or even outdoors, though the A-BEC unlockable hub electric wheelchair motor-wheel units are the very best for large mobile robots. Also note the 'necklace' of security system PIR sensors around Robart's neck which are useful for detecting people moving around the robot.

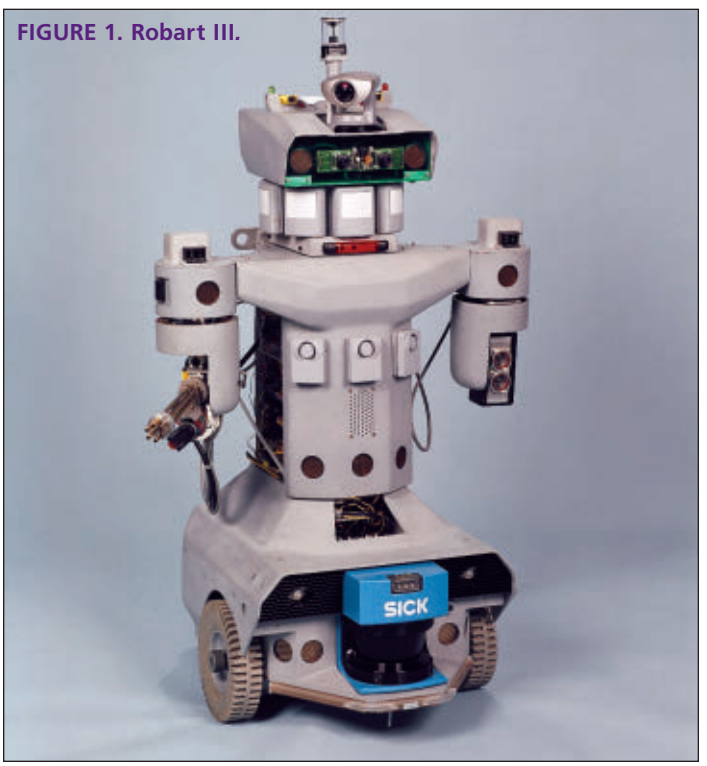


FIGURE 1. Robart III.

Passive Infrared Sensors

Last month, I concentrated on basic object detection, tactile, active IR, ultrasonic sensors, and a simple laser range finder. I specifically did not mention passive IR sensors since those are object-specific; they only sense objects that emit IR radiation at human body temperatures, and the object or the robot must be moving. A lot of people seem to feel that PIR (passive infrared) sensors detect people when, in fact, they can only detect temperature changes that a moving person causes. Therefore, they are people motion detectors. PIR sensors have a crystal of pyroelectric (heat changing electrical characteristics) material such as lithium tantalate ($LiTaO_3$), gallium nitride (GaN), cesium nitrate ($CsNO_3$), and other materials that react to a change of charge status when the

IR radiation focused upon it changes. A filter is usually placed in front of the sensor to restrict IR radiation to the 8-14 µm wavelength range, of which humans emit around 9.4 µm. The actual sensor inside looks like an old TO-3 transistor with a clear plastic window on the top that sits at the focal point of the Fresnel lenses molded into the back of the plastic.

Figure 2 shows how the lens' concentric rings — each one shaped like a circular part of a convex lens — focus the image in a small spot. The crystal or material and associated electronics can only detect a person when the person's (or pet or heating vent) focused image moves off of or on to the crystal, causing a change in the crystal's charge. If the background happens to be close to the typical 93ºF skin temperature of a moving human being, they will not be detected.

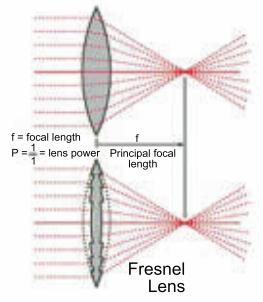


FIGURE 2. Fresnel focuses on PIR.

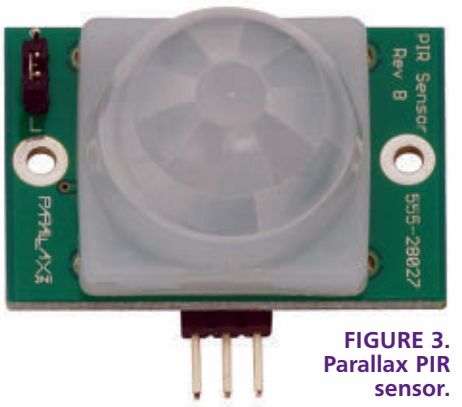


FIGURE 3. Parallax PIR sensor.

The milky-white plastic cover on most motion detectors — whether intrusion alarm types, outdoor light sensors, or robot-specific — are really a series of Fresnel high density polyethylene (HDPE) lenses that are pointed in many directions to detect an IR image and then momentarily focus each image as it scans across the PIR sensor. Normal glass does not pass infrared radiation, so you cannot place a PIR sensor behind regular glass and have it work. The cylindrically-curved sensors that are the most popular for security and light control detect motion in a plane, whereas the spherical PIR detectors popular with robot builders have a semi-spherical detection range.

Another novel method of causing a 93ºF IR heat image to rapidly appear and disappear on a PIR sensor is to use a slowly rotating 'chopper wheel' in front of the sensor, much like a toothed gear sometimes used in wheel encoders. A single HDPE plastic lens can be used to make a very narrow beam people detector. The person does not need to be moving, just the robot (or a scanned detector). Once the image is detected by this type of PIR sensor, the target or the sensor does not need to move to continually detect the presence of a human.

The Parallax PIR Sensor

Figure 3 shows a typical PIR sensor used in mobile robots. It is passive since it does not actively emit radiation; rather, it detects IR radiation. When the Parallax Rev B PIR sensor detects motion, a high logic signal is put out for reading by all microcontrollers, or even driving simple logic circuitry. Range can be selected by an onboard jumper from 15 feet to over 30 feet in the long range position of the jumper. It is a tiny unit with SMT circuit mounting; dimensions are 1.41 x 1.0 x 0.8 inches and it has a three-pin SIP header for power, ground, and signal. It operates on 5.0 VDC @ 23 mA, but can handle 3V to 6V. Onboard LEDs light up the domed lens for fast visual feedback when motion is detected, and the sensor has 2-56 holes for mounting. I tested it on a Parallax Boe-Bot and (as expected) I saw sensitivity that ranged from 10 feet to over 30 feet. The range was affected depending on how I moved, if I had a jacket on, or if it was indoors or out in my cooler garage.

This is one of the most popular robot sensors and costs $10.99 ($7.99 for the earlier Rev A). There are several other robot-specific PIR sensors handled by Acroname, Pololu, and the RobotShop, among others.

Microwave Frequency (X-Band) Motion Detector

Detecting objects by radio waves was first observed by the German physicist Heinrich Hertz in 1887, when he found that electromagnetic waves could travel through walls and other materials, and were reflected by conductors and dielectrics. The development of modern radar began long before WWII, but was put into use just at the beginning of the war. Many people know that the Japanese attack on Pearl Harbor was detected by an early Army radar, but the operators failed to realize that it was not a returning squadron of American bombers. We all know what happened next. RADAR — which stands for RAdio Detection And Ranging — is used in so many ways today. Robots can now use a similar, inexpensive product to detect the nearby motion of humans — even behind walls.

FIGURE 4. Parallax X-Band motion detector.

The Parallax 32213 X-Band motion detector shown in **Figure 4** operates in the microwave region of 10.525 GHz and can detect motions of humans, other robots, or even animals from about eight feet to 30 feet. It is more immune

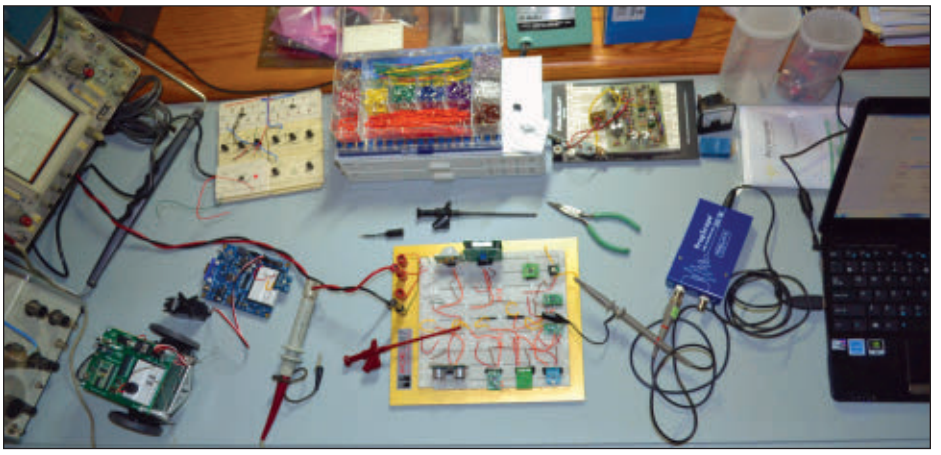


FIGURE 5. Test bench and breadboard setup.

FIGURE 6. SINS Mk2 Mod 6 submarine inertial navigation system.

to false triggering than PIR sensors, and this type of technology is often used in conjunction with a PIR sensor to verify a target or movement in security systems. There is a sensitivity trimpot on the back to set a desirable range, and it delivers a 0.0 to 3.9 volt low/high signal through its four-pin header when movement is detected. I mounted the detector on a solderless breadboard (shown in **Figure 5**) on my workbench, alongside other sensors. I used an oscilloscope to see the high/low changes when testing. It operates from a regulated 5 VDC but has a built-in series resistor for compatibility with the Propeller microcontroller and other 3.3V devices. I used two microcontroller platforms: the Propeller Board of Education (BOE), as well as a Boe-Bot that includes the BASIC Stamp BOE (also shown in **Figure 5**). I kept a thin pad of black conductive foam under each module for safety when they were not being used. The Parallax PropScope and my trusty old Tektronix 475A were used for waveform analysis. I spent more time playing with the motion detector and a Pololu MinIMU (discussed below) board than I should have.

Early Inertial Navigation Systems

Decades ago, submarines traveling below the waves for months at a time relied on what was called SINS, or Ship's Inertial Navigation Systems. A cut-away of an actual system is shown in **Figure 6**. It consists of the shock-protected binnacle in the center that contains the gyros and accelerometers; a console that contains the electronics that drive servos to keep the platform stable; and the navigation computer that tracks orientation, the ship's motions, earth's motion, and multi-axis acceleration, and computes the required course, latitude, and longitude. It also compensates for roll, pitch, and yaw data that is fed into weapon's fire control systems. Today's robot builders desiring good robot motion control can have the equivalent of this thousands of pounds of metal, transistors, and heavy gyros shrunk to a thumbnail size circuit board for just a few dollars. That's what today's MEMS and modern ICs get you.

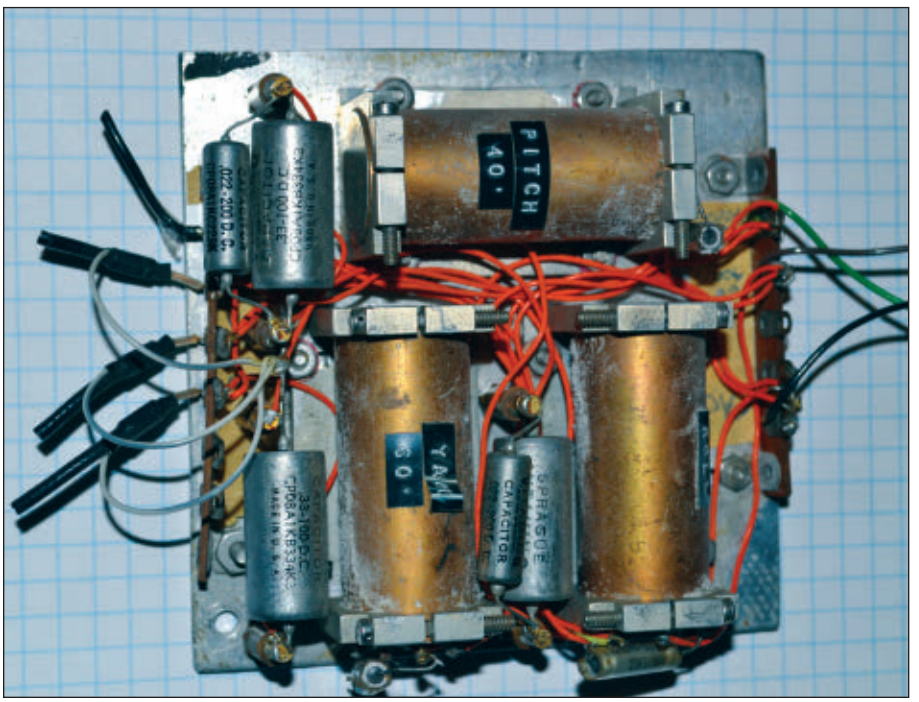


FIGURE 7. An old surplus three-axis electromechanical gyro system.

Gyros Detect the Robot's Turning Rate

Mobile robots move and encounter all sorts of acceleration and rotation during their travels. They can be made to move more accurately and follow a programmed path when using feedback from motion sensors. Gyros can detect the robot's turning rate, as well as its turning velocity. Years ago, I ran across

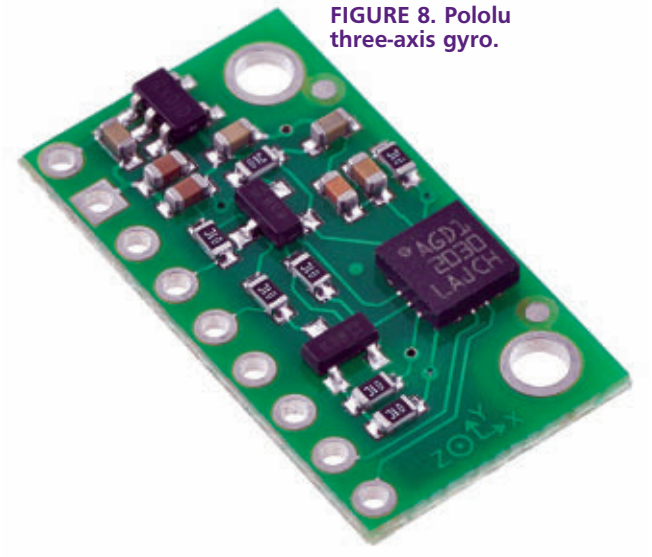


FIGURE 8. Pololu three-axis gyro.

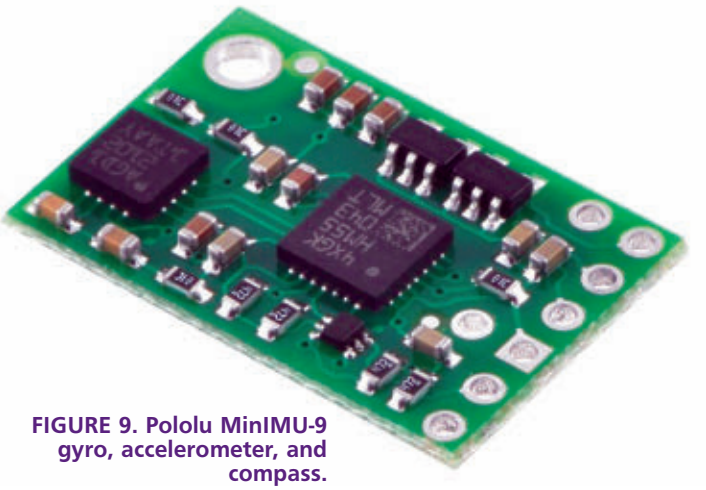


FIGURE 9. Pololu MinIMU-9 gyro, accelerometer, and compass.

the gyro board shown in **Figure 7** at a surplus sale and it cost me about $10. It was a complex piece of equipment for its time with three separate gyros for pitch, yaw, and roll, and ran off 28 volts AC @ 400 Hz. I was going to use it on a large mobile robot and got it to operate for testing purposes at work, but never implemented it in my robot. Gyros have certainly come a long way since the early '80s with so many capable application-specific MEMS (MicroElectro Mechanical Systems) chips available on tiny, low power circuit boards.

The Pololu L2G4200D shown in **Figure 8** is a popular three-axis gyro, but a few experimenters may find its small size and leadless interconnection a bit difficult to work with. As I didn't have the correct nine-pin (or more) connector for the included 0.1" pin spacing headers, I just mounted the header pins on my solderless breadboard and the hookup was simple. The ST L3G4200D measures the angular rates of rotation about the pitch (x), roll (y), and yaw (z) axes. Angular velocity measurements with a configurable range of ±250°/s, ±500°/s, or ±2,000°/s can be read through a digital I²C or SPI interface. The IC is a 3.3V device, but an onboard voltage regulator allows operation from 2.5 to 5.5 volts @ 7.0 mA — perfect for all popular microcontrollers. This device is a perfect low cost gyro system for any robot.

Adding a Bit More to Determine All the Robot Movements

Going a bit further and adding a three-axis accelerometer and a three-axis magnetometer compass to a three-axis gyro, you have the Pololu MinIMU #1265 shown in **Figure 9**. This inertial measurement unit has the same L2G4200D gyro mentioned above, but adds an LSM303DLM accelerometer chip and three-axis magnetometer onto a 0.9" x 0.6" leadless board. The nine independent rotation, acceleration, and magnetic measurements can be used to calculate the sensor's absolute orientation. Operating on 2.6V to 5.5V @ 10 mA, this tiny board replaces what cost $100K or more on missiles from 30 years ago.

The output format is one 16-bit reading per axis for the gyro; a 12-bit reading (left-justified) for the accelerometer; and a 12-bit reading (right-justified) for the magnetometer compass. The sensitivity of the gyro can be adjusted from ±250 to ±2,000°/s; ±2 to ±8 gs for the accelerometer; and ±1.3 to ±8.1 gauss for the magnetometer. The Pololu 3D compass and accelerometer shown in **Figure 10** (with similar physical, electrical, and mechanical specifications) may be what you would need for the navigation core of your latest robot. For strictly three-axis gyro information, the L3G4200D is a great choice with identical voltage and mounting and signal characteristics.

All of the Pololu sensor modules are leadless, yet come with a set of two headers, a set of straight pins, and a 90º set of pins. They are tiny and a bit difficult to handle but are perfect for small as well as large robots, and for quad-rotor autonomous flying vehicles. I highly recommend

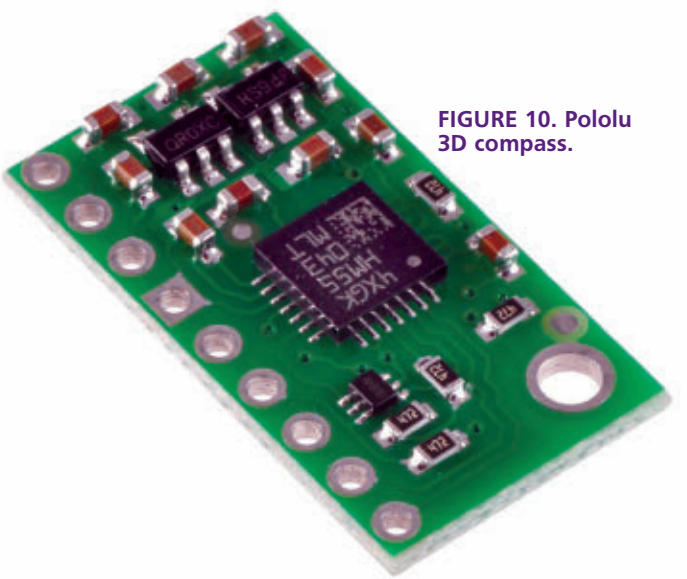


FIGURE 10. Pololu 3D compass.

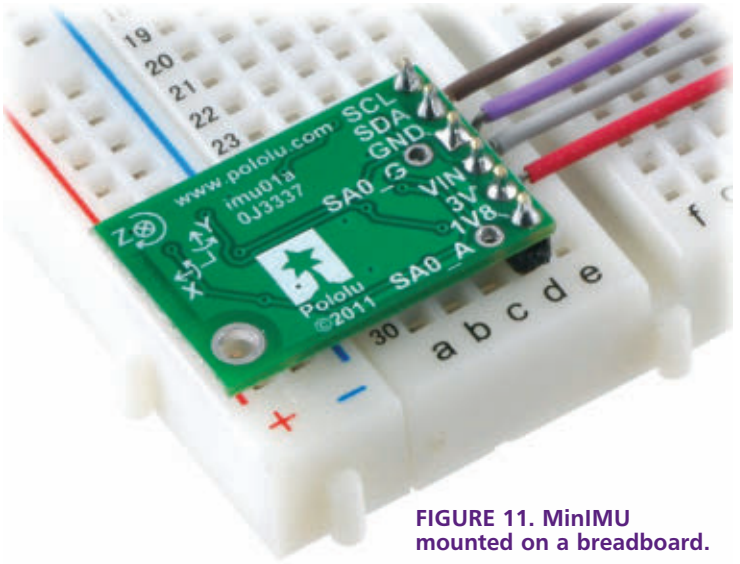


FIGURE 11. MinIMU mounted on a breadboard.

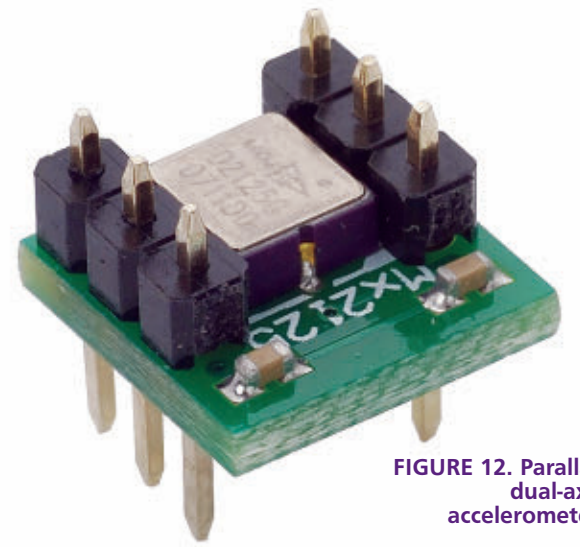


FIGURE 12. Parallax dual-axis accelerometer.

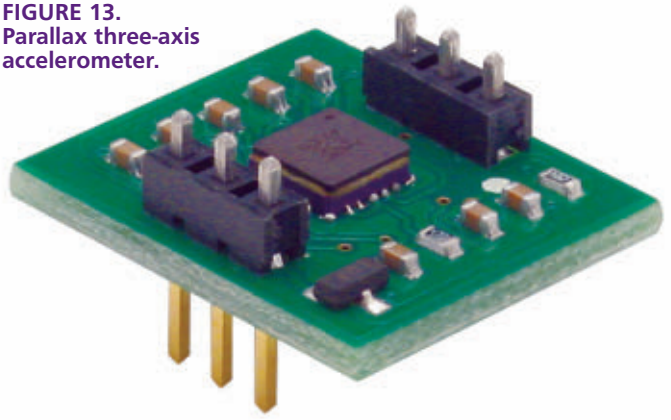
reading all of the applicable literature available on the Pololu website, as well as all the other company websites. There are hundreds of pages, and some of that material may be skimmed over. The interfacing, mounting, specifications, register description, and a raft of very descriptive tables are available for each chip.

I had a bit of trouble with testing the L2G4200D device; not because it was defective in any way, but because one of the jumper wires on my solderless breadboard was making intermittent contact as I moved the board in different orientations with my hands. Unfortunately, that happened several times with different modules before I discovered the problem causing the error messages. It was the shaking of the board to test the accelerometers that loosened the wire intermittently. Use your breadboard for initial testing, but securely plug in or solder the board for your finished robot. **Figure 11** shows a good way to secure a module on a breadboard for testing using a straight six-pin header. Note the X-Y-Z orientation markings on the back of the IMU board.

If you're looking for an incredibly small dual-axis accelerometer, the Parallax 0.42" square Memsic 2125 is a great choice. The low cost, dual-axis thermal accelerometer shown in **Figure 12** is capable of measuring tilt, acceleration, rotation, and vibration with a range of ±3 g, and is electrically compatible with other popular accelerometers. Its six pins spaced at 0.1" work great for solderless breadboards such as the Boe-Bot or the many experimental platforms such as the Propeller-based PropBOE and Arduino boards. Operating from 3.3 or 5 VDC @ 4.0 mA, it outputs a TTL compatible 100 Hz PWM signal proportional to acceleration, and is a great choice for autonomous robotics applications.

Parallax also has a three-axis accelerometer module (the MMA7455) shown in **Figure 13**. Utilizing the Freescale Semiconductor MMA7455L MEMS chip, it operates from 2.5V to 5.5V with a very low 26 µA in standby, and less than 3 mils when operating. It features an ADC, digital low pass filter, and selectable sensitivity ranges of ±2 g, ±4 g, or ±8 g. This device can easily be configured to detect quick motion pulses as single taps, double taps, and 0 g free-fall conditions. Communicating via an eight-bit Serial Peripheral Interface (SPI) or Inter-Integrated Circuit (I^2C) bus at a max of 8 MHz, it worked well with my Propeller setup and should interface with BASIC Stamps or any of the Arduino series. Quad rotor or any mobile robots should find this inexpensive module useful for three-axis motion sensing, as well as vibration analysis.

FIGURE 13. Parallax three-axis accelerometer.



Robot Compasses

Magnetic compasses have been and still are an excellent way for people, robots, planes, and many other moving vehicles to determine their direction in relation to the earth's magnetic field. However, there are a few hiccups in using a magnetic compass and knowing which direction you are facing. Sailors have long known that the needle or "N" on a compass rarely points north as the geographic North Pole is nowhere near the magnetic north pole that is in northern Canada. For example, you could be in northern Alaska facing true north and your compass needle could be pointing 90º to your right. There is a wavy

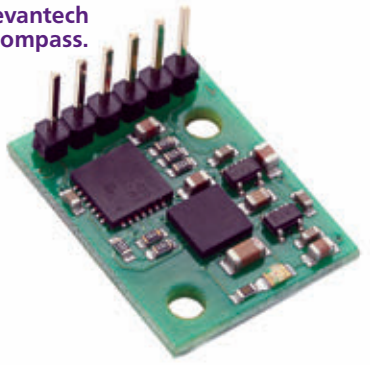


FIGURE 14. Acroname Devantech R351-CMPS10 compass.

line running north and south in the Central US where a magnetic compass will actually point to the north and that line has zero 'variation.' In the northwest, the compass will point a bit to the east, and the opposite direction in the northeast.

Boats — as well as robots — also have another magnetic problem called 'deviation' that is caused by electrical fields, ferrous materials, and magnetic items, and correcting these problems can be as simple as carefully placing a few tiny magnets near your compass. Nautical charts have variations actually stated on them. Sailors and robot builders can use the chart in **Table 1** to correct for such discrepancies to arrive at a compass or true course.

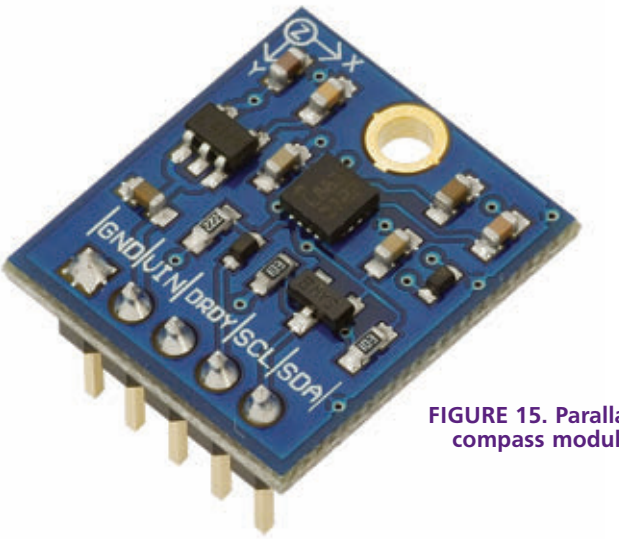


FIGURE 15. Parallax compass module.

Compass	Deviation	Magnetic	Variation	True
358	5E	003	6E	009
True	Variation	Magnetic	Deviation	Compass
009	6E	003	5E	358

Table 1.

Variation and Deviation Correction Examples

Magnetic compass corrections involve a bit of simple calculating that are important. When I took a few US Power Squadron courses years ago, we used these two mnemonics to remember the right five words for the order of the compass correction words: "Can Dead Men Vote Twice;" and "True Virgins Make Dull Companions." (Hey, it worked.) Variation in robots can be caused by electrical wires or motors turning on and off, so proper compass placement as well as shielding is sometimes required. Most of the time, robot builders will never have to use these corrections.

The Acroname Devantech R351-CMPS10 compass shown in **Figure 14** is an example of a popular robot experimenter's course-finding module. It is affordable and according to the website, features a three-axis magnetometer and a three-axis accelerometer to compensate for up to 60 degrees of tilt. Module data is accessible via serial, I²C, and as a PWM output. In addition to a bearing reading with 0.1º resolution, this compass provides pitch, roll, and yaw accelerometer and magnetometer readings. Operating from 3.3 to 5.0 volts @ 25 mA, the small 11/16" x 15/16" board outputs a serial 9600 baud, no parity, two stop bits with 3.3V to 5V signal levels, or a SMBUS compatible at a 100 kHz clock rate.

The Parallax three-axis compass module shown in **Figure 15** has the Honeywell HMC5883L magneto-resistive sensor circuit that utilizes three discrete sensors to measure magnetic fields. It comes as a six-pin package with 0.1" spacing on a 0.725" x 0.650" board that is easily mounted on a breadboard or hardwired into a permanent circuit. Operating from 2.7V to 6.5V, it has a resolution of one to two degrees within a ±8 gauss range. The communication interface is I²C, up to 400 kHz.

As with any magnetic compass, be careful where and how you mount the circuit as ferrous screws and brackets can interfere with the accuracy, and actual readings may be different than when testing the circuit on a breadboard. I recommend the use of nylon screws and nuts; do a final test with the circuit in the completed robot.

Final Thoughts

I've just touched on a few of the motion and orientation detecting sensors that are popular with experimental mobile robots. Needless to say, odometry via shaft encoders on the robot's wheels and motor shafts are an important part of accurately determining how far and fast a robot is moving; I decided not to include the simple encoders and emitter/receptor pairs in the sensor section.

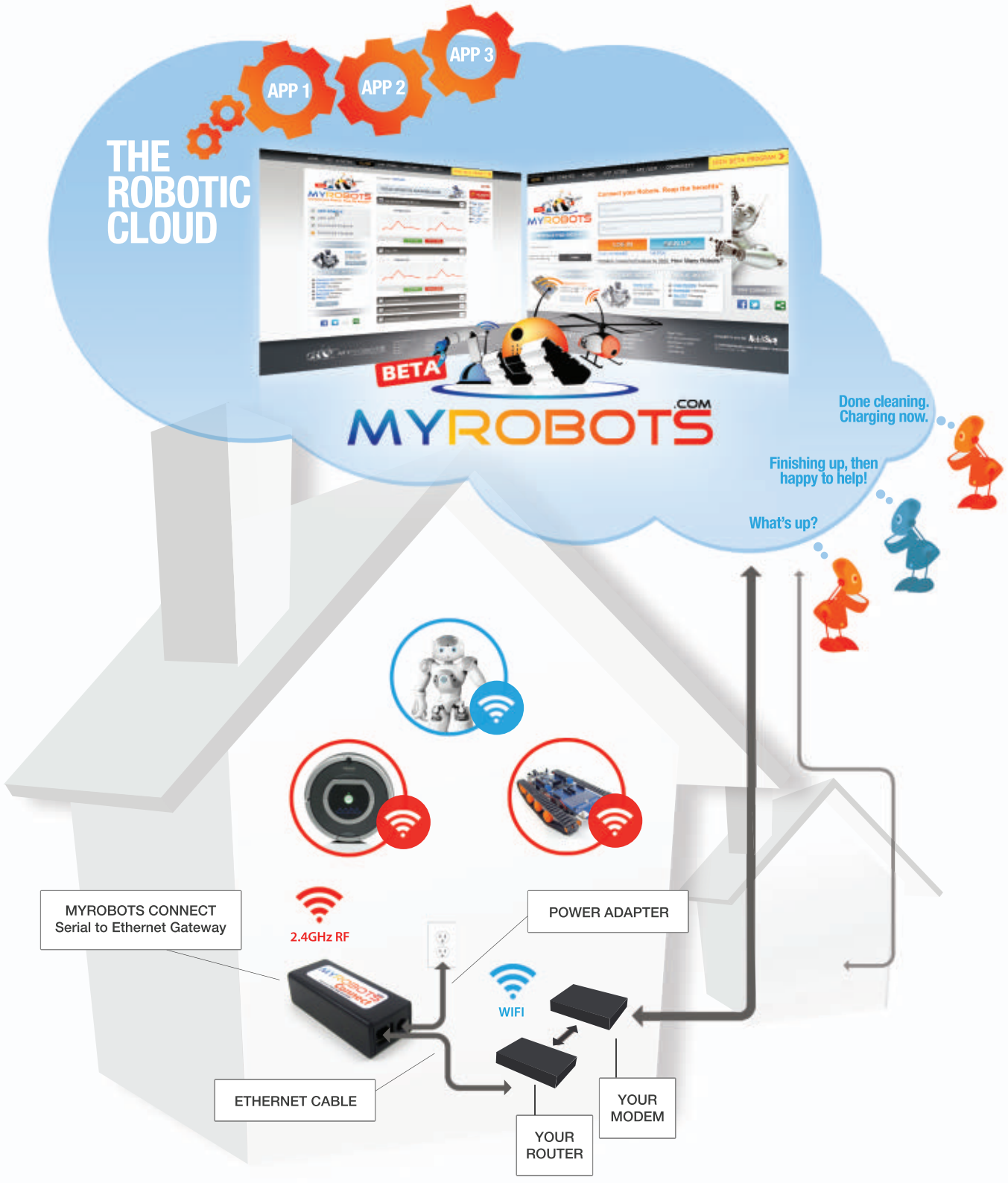
Next month, I will touch on some of the more unique and less-used sensors for robots such as GPS localization, gas detection, force, light, color, and then will end with the latest Kinect from Microsoft (designed for Windows applications). Many thousands of sensors are available to detect every type of phenomena, force, position, radiation, and anything you can imagine, and are the most important additions to any robot. Check out the myriad of types offered by the manufacturers advertising in this magazine to make your robot a bit more capable in sensing the world around it. As always, I appreciate the emails and calls concerning corrections, ideas, and topics that you would like to see in my column. **SV**

Tom Carroll can be reached at TWCarroll@aol.com.

ROBO-LINKS

For the finest in robots, parts, and services, go to www.servomagazine.com and click on Robo-Links.

Advertiser Index



Connect your robots. Reap the benefits ™

MyRobots.com strives to make cloud robotics a reality accessible to everyone and everything by enabling all robots and smart devices to connect to the Internet.

You can think of MyRobots.com as a social network for robots and smart devices. In the same way humans benefit from socializing, collaborating and sharing, robots can benefit from interacting and sharing their sensors' information, which provides insight on their current state, and allows them to be controlled and monitored remotely.

Through the **MyRobots App Store**, robots augment their capabilities so they can transcend their limitations, allowing them to do more than what they were originally designed for. This means robot owners can reap the full benefits of their robots.

Open Platform Humanoid Project

DARwIn-OP

Open Platform

- All PC-based sources open (works on Linux Ubuntu)
- Sub board circuit & firmware open
- Physical specifications of H/W and 3D modeling data open
- Peripheral expansion (13×GP I/O&ADC)

High Performance

- Default walking speed : 24cm/sec (9.5 in/sec) – user modifiable gait
- Built-in PC : 1.6 GHz Intel Atom Z530 on-board 4GB SSD
- 2MP USB Camera, 3-axis gyro, 3-axis accelerometer

Easy Maintenance

- Single type actuators with durable metallic gears
- Modular Structure
- Part replacements can be easily made by user

Team DARwIn: RoboCup 2011 World Champion

Specification (20 DOF)

- Weight : 2.9 kgs (6.39 lbs)
- Height : 454.5 mm (17.90 inches)
- For more information, please visit www.robotsource.org

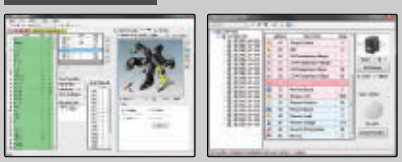
RELEASED RS485

DYNAMIXEL MX-Series

MX-28T / MX-28R | MX-64T / MX-64R | MX-106T / MX-106R

Supports TTL and RS485.

S/W Tools



MATLAB® python ROS Visual Studio LabVIEW C/C++ C# Visual Basic Java

Contactless Absolute Encoder

12-bit resolution, 360° operating range with super-durability

User Programmable PID Control Gain

Precise and reliable PID control

Powerful Resource & Organized Design

Maxon motor, 32 bit controller, high communication speed
RX and EX series compatible dimension

Current Based Torque Control

Position and speed control with dual-loop current control
(MX-64, MX-106 only)

[USA] E-mail : america@robotis.com Tel : +1-949-333-3635
[KOREA] E-mail : korea@robotis.com Tel : +82-70-8671-2600
[JAPAN] E-mail : japan@robotis.com Tel : +81-3-4330-3660
[OTHERS] E-mail : contactus2@robotis.com Tel : +82-70-8671-2609

ROBOTIS
www.robotis.com

SERVOCITY

We have the parts for your

SERVOCITY™

YouTube | youtube.com/ServoCity
Facebook | facebook.com/ServoCity
Twitter | @ServoCity

one online store | *thousands* of products

millions of possibilites

To view our entire selection of products visit us online at

www.SERVOCITY.com

Specializing in servos, radios, motors, pan & tilt systems, batteries, wiring, robot kits, hardware, electronics, actuators, gearboxes, and much more!

Life with Robot!
Not a dream anymore

THE MASTERPIECE
of ROBOT SERVO

24kgf.cm @ 7.4V
[333.6 ozf.in.]

Developed with the ease of use in mind, Herkulex Smart Servo Series are the best 'Green' energy efficient smart servos in the world with low voltage, high output, asynchronous bi-directional communication, and more than 50 set-up parameters.

HerkuleX

Specification (DRS-0101, DRS-0201)

- **Dimensions** : 44.5mm(W) X 24.0mm(D) X 32.0mm(H) [1.75in. X 0.94in. X 1.26in.]
- **Weight** : 45g **(DRS-0101)** / 60g **(DRS-0201)** [1.59oz **(DRS-0101)** / 2.12oz **(DRS-0201)**]
- **Input Voltage** : 7.4V DC **(DRS-0101)** / 7~12V DC (Optimized 7.4V) **(DRS-0201)**
- **Stall Torque** : 12kgf.cm@7.4V **(DRS-0101)** / 24kgf.cm@7.4V **(DRS-0201)**
  [166.8 ozf.in. **(DRS-0101)** / 333.6 ozf.in. **(DRS-0201)**]
- **Maximum Speed** : 0.166s/60 ° @7.4V **(DRS-0101)** / 0.147s/60 ° @7.4V **(DRS-0201)**
- **Operating Angle** : 320 °, Continuous Rotation
- **Communication** : Full Duplex Asynchronous Serial(TTL), Multi Drop, 0-254 ID, Maximum Baud Rate : 0.67Mbps
- **Motor** : Metal Brush DC Cored **(DRS-0101)** / Coreless DC **(DRS-0201)**
- **Gear** : Super Engineering Plastic **(DRS-0101)** / Reinforced Metal **(DRS-0201)**
- **Feedback** : Position, Speed, Temperature, Load, Voltage, etc.
- **Features** : PID, Feedforward, Trapezoidal Velocity Profile, Velocity Override, Torque Saturator & Offset, Overload Protection, Neutral Calibration, Deadband, 54 Selectable Setting Parameters (Sold Separately : HerkuleX Manager Kit)

HerkuleX Manager is a bundled software that uses the GUI to maximize the ease of operation in setting up more than 50 servo operating parameters and servo maintenance using such a tool as the real time trend graph.

HOVIS Lite HOVIS Genie HOVIS Eco

Dongbu Robot

Dongbu Robot Co., Ltd. www.dongburobot.com

Headquarter : 11th Floor, Bucheon Techno Park 401, Yakdae-dong, Wonmi-gu, Bucheon-city, Gyunggi-do 420-734, Korea
TEL : +82-32-329-5551(ext.311), FAX : +82-32-329-5569, E-MAIL : robotsales@dongburobot.com
Factory : 27, 6 Gil, 4 Sandan, Jiksan-eup, Seobuk-gu, Cheonan-city, Chungcheongnam-do 331-814, Korea, TEL : +82-41-590-1700, FAX : +82-41-590-1701